A New Rule Scheduling Approach based on Estimation of Rule Execution Probability in Active Database

Abbas Rasoolzadegan[1], Rohollah Alesheykh[2], Ahmad Abdollahzadeh[1]

[1]Intelligent Systems Laboratory http://ceit.aut.ac.ir/islab
Department of Computer Engineering and Information Technology
Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran
[2]Faculty of Engineering, Payame Noor University (PNU)
rasoolzadegan@yahoo.com, alesheykh@pnu.ac.ir, *Ahmad@ceit.aut.ac.ir*

Abstract

Active database systems (ADBS) can, automatically, react to the occurrence of predefined events by definition a collection of active rules. One of the most important modules of ADBS is the rule scheduler, which has considerable impact on performance and efficiency of ADBS. Rule scheduler selects a rule to execute in each time through the rules, which are ready for execution. We have already evaluated and compared the existing rule scheduling approaches in a laboratorial environment based on three tier architecture. Five evaluation criteria were, formally, recognized and defined for evaluation and comparison of rule scheduling approaches including: Average Response Time, Response Time Standard Deviation, Throughput, Time Overhead per Transaction and CPU Utilization. At last, we introduced the most effective approach. In this paper, we propose a new approach to improve the rule scheduling based on improvement of Rule Execution Probability Estimation. Then, we compare it with the most effective existing approach in the framework mentioned. Results of experiments show that the new method improves the rule scheduling.

Keywords

Estimation of Rule Execution Probability, Active Database Management Systems, Rule Scheduling

1. Introduction

Common (Traditional) database systems often have passive nature. It means operations such as: querying, updating, inserting, deleting, reporting, and etc are performed just provided that users request them. So database management system can't automatically react when special situations occur in the system. Many applications such as warehousing programs, automation of factories, systems with financial sophisticated calculations (e.g. stock market), and etc need automatic supervision to react appropriately when predefined events occur. For supporting this reactive behavior, a new database system has been designed and called Active Database Systems (ADBS). Reactive behavior of ADBS is organized by creating Event-Condition-Action (ECA) rules (active rules). An ECA rule has three main sections: Event, Condition, and Action. When the event occurs, condition gets evaluated and if the condition is true, the action is executed. Below, you can see a simple example of ECA rule defined in an active database system for buying and selling stocks:

DEFINE *LowRisk*
ON *Stock.UpdatePrice*
IF *(Stock.policy = Low_risk) and*
*(Stock.price < Stock.initprice * e) ;(0<e<1)*
DO *Stock.Buy*

First of all, in ADBS at first the application runs until an event occurs, then the rule processing unit is activated and triggers the appropriate rule(s). Triggered rule(s) are queued in a temporary buffer. Then triggered rule(s) are successively selected according to some special criteria for the evaluation of condition. If condition is true, the action section will be executed. The executed rule may trigger some other rules subsequently; new triggered rules will be passed to the rule processing unit. When there aren't any triggered rules, the application continues running. The operations set mentioned above is called rules processing cycle. In summary, there are five different rule processing steps:

1. Event Signaling: When a primitive event occurs, the primitive event detector signals it. Additionally, the composite event detector

considers these primitive events contributed to composite events.

2. Rule Triggering: After an event is signaled, ECA rules that correspond to the event signaled are selected, and for each of them rule instances are created. In each rule instance, there is some additional information based on scheduling approach, such as timestamp, deadline, execution time, etc. These rule instances are buffered to use in the next step.

3. Condition Evaluation: After buffering rule instances, their conditions are evaluated. Then, for each rule with a true condition, a transaction is generated according to its action section.

4. Transaction Selection: This step is also called transaction scheduling phase. In this phase, a selection algorithm [6] operates on execution buffer and selects one transaction which is generated based on triggered rules, and sends the transaction to the execution unit.

5. Transaction Execution: Transactions generated based on triggered rules are executed in this phase.

One of the most important features that affects the rule processing phases a lot and plays an important role in the specification of ECA rules are coupling modes. The phases of rule processing discussed so far are not necessarily executed contiguously, but depend on the so-called coupling modes which are pairs of values (x, y) associated with each rule. The value 'x' couples event signaling and condition evaluation of a rule, whereas 'y' couples condition evaluation and action execution. Possible coupling modes are immediate, deferred and independent [4]:

- Immediate mode: in this mode, when an event occurs, corresponding rule is triggered, then current transaction is suspended and action section of the triggered rule is executed, if the condition holds.
- Deferred mode: in this mode, after the occurrence of an event condition evaluation and action execution of the triggered rule is deferred till the end of the current transaction. In deferred mode, the action of triggered rule should be executed before current transaction commits.
- Independent mode: when an event occurs in independent mode, there are no time-constraints and restrictions on condition evaluation and action execution of the triggered rule.

The approach used for rule scheduling has great and direct effect on some criteria such as Average Response Time, Throughput and generally on ADBS performance. One of the weak points of ADBS is the rule scheduling approaches which have already been presented. Some of existing approaches were designed for special situations and the rest of them don't have enough performance and efficiency. Rule scheduling approaches in ADBS is an important research topic.

This paper has five sections. In section two, we analyze existing rule scheduling approaches in ADBS. In section three, we introduce a framework to compare and evaluate existing rule scheduling approaches. In this framework, five evaluation criteria have been proposed: Average Response Time, Response Time Standard Deviation, Throughput, Time Overhead per Transaction and CPU Utilization. At the end of this section the approach which has the most positive impact on performance and efficiency of ADBS has been introduced by analyzing the weaknesses and strengths of existing approaches. Then, in section four we introduce a new algorithm for Estimation of Rule Execution Probability and develop a new scheduling approach based on it. Then we show the positive impact of this algorithm on performance of ADBS. Finally, in section five, there is a conclusion of subjects presented in this paper.

2. Existing Rule Scheduling Approaches

In ADBS, the process of priority allocation to rules, ready for execution, is called rule scheduling. As we have also mentioned before, a rule gets ready for execution if and only if, firstly, get activated because of occurrence of the corresponding event in the system and secondly, its condition seems true in evaluation time. In this section, we briefly describe the approaches used for rules scheduling. In this paper, we use "rule" and "transaction" terms interchangeably.

2.1 Random Scheduling Approach

Random selection is one of the easiest approaches for rule scheduling in ADBS. This approach has been implemented in RPL and Ode active database systems [6]. In the Random approach ADBMS selects randomly one of the activated rules. The most important characteristic of this approach is its simplicity, at the cost of efficiency.

2.2 Static Priority Scheduling Approach

In this approach, the system assigns a numeric priority to each ECA rule but the priorities need not be unique. In the Ariel and POSTGRES systems, each rule is assigned a priority between -1000 and +1000. When an activated rule should be selected to run, the rule that has the minimum static priority is selected [6].

2.3 FCFS Scheduling Approach

FCFS (First Come First Serve) scheduling approach is one of the classic approaches used for rule scheduling in ADBS [6]. When an event occurs and rules are triggered, an instance of each triggered rule is generated. This instance of triggered rule contains a timestamp which shows the time when the rule is triggered. When an activated rule should be selected to run, the activated rule that has the earliest timestamp is selected. This scheduling approach is used in SAMOS.

2.4 EDF based Scheduling Approach

Earliest Deadline First (EDF) is one of the classic algorithms for transaction scheduling in real-time systems [6]. The EDF based approach is one of the best approaches introduced for rule scheduling till now. This approach has been presented for Real-time Active Database (RADB). In this approach rules are scheduled based on their deadline. This approach has three different versions: (1) EDF_{PD}, (2) EDF_{DIV}, and (3) EDF_{SL}. The EDF_{PD} is a static baseline policy where the rules priorities do not change with time. EDF_{DIV} and EDF_{SL} are dynamic policies where rules priorities change depending on the amount of dynamic work they have generated [1].

2.5 E_x-SJF Scheduling Approach

This method is based on Shortest Job First algorithm. The SJF algorithm is one of the most effective classic scheduling approaches [5]. This algorithm is not useful for rule scheduling in ADBS due to active work load nature [1] of it. So there is defined preprocesses for preparing rule base to use the SJF algorithm for rule scheduling in E_x-SJF (Extended SJF) approach [5]. The difference between SJF and E_x-SJF is in manner of transactions (rules) execution time calculation. In E_x-SJF approach, the execution time of each rule (parent) is related to the number of its immediate and deferred child rules. According to the manner of interference of the execution time of immediate and deferred child rules in the execution time of their parent rules, there are three versions of E_x-SJF which are named E_x-SJF_{EXA}, E_x-SJF_{PRO}, and E_x-SJF_{PRO}-V.1.8 [5].

Although calculation of rules execution time is possible in run time, it leads to an inefficient scheduling approach because of its too much time overhead. So all versions of E_x-SJF calculate execution time of the rules before system's run time. As we mentioned before, executing of active rules depend on condition evaluation result of those rules in evaluation time. In other words, execution probability of the activated rules equals condition correctness probability of those rules. So we should estimate execution probability of the activated rules before system's run time. More precise estimation of rule execution probability leads to more precise calculation of rule execution time [5].

The condition section of the rules consists of conditional expressions, database query statements, recalling the procedures, functions, and logical composition of them. Suppose that condition section of rule R is defined like $[(A \cap B) \cup (C \cap D)]$. A, B, C, and D are logical statements. So correctness probability of condition of R is calculated as below:

$$P[(A\cap B)\cup(C\cap D)]=P(A\cap B)+P(C\cap D)-P(A\cap B\cap C\cap D)$$

Supposing that A, B, C, and D are independent from each other, we will have:

$$P[(A\cap B)\cup(C\cap D)]=P(A)*P(B)+P(C)*P(D)-P(A)*P(B)*P(C)*P(D) \quad (1)$$

If correctness probability of logical statements A, B, C, and D exist definitely, with putting their values in relation (1), we can calculate the correctness probability of condition of R precisely. But correctness probability of a conditional statement often doesn't exist before its execution. So we should estimate correctness probability of conditions before system's run time.

2.5.1 E_x-SJF_{EXA} Scheduling Approach

In this approach condition correctness probability of each rule (rule execution probability) is considered 1 [1]. So rules execution time is calculated by relation (2) supposing $P(R) = 1$.

$$X(R)=L(R)+\sum_{i=1}^{n^{imm}(R)}P(R_i)*X^{imm}(R_i)+\sum_{j=1}^{n^{def}(R)}P(R_j)*X^{def}(R_j) \quad (2)$$

$P(R_J)$	The correctness probability of condition section of deferred rule Rj activated by R
$P(R_I)$	The correctness probability of condition section of immediate rule R_i activated by R
$X^{imm}(R_i)$	The execution time of rule R_i triggered in immediate mode
$X^{def}(R_j)$	The execution time of rule R_j triggered in deferred mode
$L(R)$	The primary execution time of rule R
$n^{def}(R)$	Number of deferred rules triggered by R during its execution
$n^{imm}(R)$	Number of immediate rules triggered by R during its execution
$X(R)$	The execution time of rule R calculated by this approach

2.5.2 E_x-SJF_{PRO} Scheduling Approach

In this approach, correctness probability of each conditional statement in the condition section of rules is equally considered 0.5. So correctness probability of the condition of R mentioned in last section according to relation (1) will become:

According to the above matters, execution time of each rule in this approach is also calculated by relation (2).

$$P(R) = P[(A \cap B) \cup (C \cap D)] = \frac{1}{2}*\frac{1}{2}+\frac{1}{2}*\frac{1}{2}-\frac{1}{2}*\frac{1}{2}*\frac{1}{2}*\frac{1}{2} = \frac{7}{16}$$

2.5.3 E_x-SJF_{PRO}-V.1.8 Scheduling Approach

In this approach, at first, execution time of each rule is calculated like E_x-SJF_{PRO}. Then in every time which rule R_1 with the shortest execution time among activated rules is selected for condition evaluation, correctness probability of each logical statement of condition section of R_1, i.e. $P(R,LS_i)$ is repeatedly calculated and stored based on relation (3).

$$P(R, LS_i) = \frac{T1}{T2} \qquad (3)$$

(T_1 shows the frequency that LS_i has been evaluated and has been true so far and T_2 shows the frequency that R has been activated so far.)

This process is repeated for each logical statement of condition section of the corresponding rule until the changes rate of correctness probability of that logical statement, achieves to a desired value (e.g., 0.0000001 and in general mode ε). At this time, new value is replaced with primary default value (i.e. ½). It is evident that the smaller value ε, the more exact calculation of $P(R,LS_i)$. When correctness probabilities of all logical statements of a condition are updated, correctness probability of that condition is updated, too. And ultimately execution time of rule R is updated, when condition section correctness probability and all R's childes execution time are updated. So, after passing a short time from start of executing the system (which this time is very insignificant in compare with total executing time of system), execution time of all rules are calculated with a satisfied precision (which amount of this precision is depend on value ε).

3. Evaluation and comparison of existing rule scheduling approaches

In reference [1] a framework is introduced for comparison and evaluation of existing rule scheduling methods. This framework contains five evaluation criteria: Average Response Time, Response Time Standard Deviation, Throughput, Time Overhead per Transaction and CPU Utilization. Table (1) defines these parameters formally.

Table 1. Formal definition of evaluation parameters

N = Number of Executed Rules ART= Average Response Time RTSV= Response Time Standard Variance U_{CPU} = CPU Utilization TOPT = Time Overhead Per Transaction	
T_1^i = Activation Time of i^{th} Rule T_2^i = Start of Execution Time of i^{th} Rule $T = (T_2^N + ExecutionTime\ of\ N^{th}\ Rule) - T_1^1$ $T^* = \sum_{i=1}^{N} Real\ Execution\ Time\ of\ i^{th}\ Rule$	
$U_{CPU} = \frac{T^*}{T}*100$	$RTSV=\sqrt{\frac{\sum_{i=1}^{N}((T_2^i - T_1^i) - ART)^2}{N}}$
$ART = \frac{\sum_{i=1}^{N}(T_2^i - T_1^i)}{N}$	Throughput $= \frac{N}{T}$ $TOPT = \frac{T - T^*}{N}$

In this framework a laboratorial environment named Active Database System Simulator (ADSS) is designed and implemented to simulate the active database system behavior. So we can implement each rule scheduling approach and consider the performance of it in ADSS. Architecture, the manner of designing and implementation of ADSS and rule scheduling approaches are extensively described in reference [1]. Figure (1) illustrates the architecture of the ADSS. An important characteristic of ADSS is flexibility. It means that we can implement each rule scheduling approach, only by replacing the scheduling algorithm in the ADSS.

The ADSS has three tier architecture: "object manager unit", "rule manager unit" and "transaction manager unit" [1]. Experiments are performed in three modes: "Deferred mode", "Immediate mode", and "Composite mode". In the first mode system uses rules only in deferred mode. In the second mode, system uses rules only in immediate mode and ultimately in the third mode, system uses rules in all immediate, deferred, and independent modes [2].

Results of experiments in deferred, immediate and composite modes are shown in tables (2), (3), (4), respectively. The content of each cell shows the rank of corresponding scheduling method according to corresponding evaluation criteria.

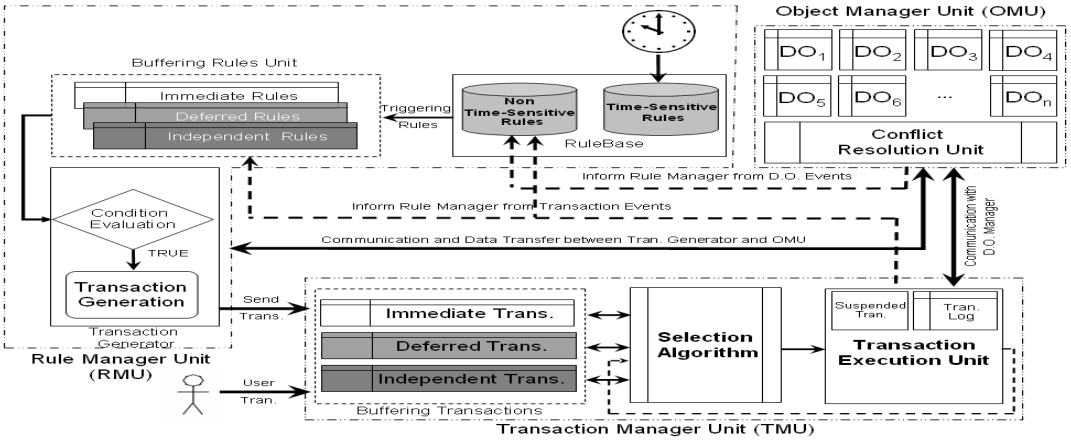

Figure 1. The Active Database System Simulator Architecture

Table 2. Results of simulation of available rule scheduling approaches in deferred mode

Methods \ Evaluation Criteria	Average Response Time	Response Time Standard Deviation	Throughput	Time Overhead per Transaction	CPU Utilization
Random	3	4	1	1	3
Static Priority	3	4	1	1	3
FCFS	3	3	1	1	3
EDF_{PD}	3	6	3	1	3
EDF_{DIV}	3	3	3	1	3
EDF_{SL}	3	5	4	1	1
E_x-SJF_{EXA}	2	5	3	1	2
E_x-SJF_{PRO}	2	2	3	1	2
E_x-SJF_{PRO}-V.1.8	1	1	2	1	2

Results of experiments show that E_x-SJF_{PRO}-V.1.8 has generally the most positive impact on performance (Average Response Time, Response Time Standard Deviation, and Throughput) and efficiency (Time Overhead per Transaction and CPU Utilization) of ADBS. In other words rule execution time is calculated in E_x-SJF_{PRO}-V.1.8 approach (by adding an estimation module) more precise than last versions of E_x-SJF approach and this subject finally leads to improvement of rule scheduling process. Process of estimation of rule execution probability doesn't impose computational overhead on system so doesn't have negative impact on Time Overhead per Transaction and CPU Utilization [5].

According to the obtained results, we are going to present a new version of Ex-SJF approach in the next section by applying a more precise technique of estimation of rule execution probability to improve scheduling process more than this

Table 3. Results of simulation of available rule scheduling approaches in immediate mode

Methods \ Evaluation Criteria	Average Response Time	Response Time Standard Deviation	Throughput	Time Overhead per Transaction	CPU Utilization
Random	4	4	2	3	3
Static Priority	4	4	2	3	3
FCFS	4	4	2	1	1
EDF_{PD}	3	3	1	1	2
EDF_{DIV}	3	3	1	1	2
EDF_{SL}	2	2	3	2	2
E_x-SJF_{EXA}	3	3	3	3	2
E_x-SJF_{PRO}	2	2	5	2	2
E_x-SJF_{PRO}-V.1.8	1	1	4	2	2

Table 4. Results of simulation of available rule scheduling approaches in composite mode

Methods \ Evaluation Criteria	Average Response Time	Response Time Standard Deviation	Throughput	Time Overhead per Transaction	CPU Utilization
Random	5	4	1	1	1
Static Priority	4	3	4	2	1
FCFS	4	3	4	2	1
EDF_{PD}	3	2	5	2	1
EDF_{DIV}	3	2	5	2	1
EDF_{SL}	3	5	6	2	1
E_x-SJF_{EXA}	2	2	3	2	1
E_x-SJF_{PRO}	2	2	3	2	1
E_x-SJF_{PRO}-V.1.8	1	1	2	2	1

.

4. E_x-SJF_{PRO}-V.2.8 as New Proposed Rule Scheduling Approach

Both E_X-SJF_{PRO}-V.2.8 and E_X-SJF_{PRO}-V.1.8 are designed and implemented based on estimation of rule

execution probability. The difference of these two approaches is in the manner of estimating [3,4,5]. In this section we are going to illustrate the operation of new technique. As we mentioned before, condition section of each rule is composed of some logical statements. Table (5) shows the condition section of some rules and table (6) shows the domain of the conditional variables of condition section of the rules mentioned in table (5). For example condition section of rule R_1 is true if data item A (DI_A) is greater than data item B (DI_B).

Table 5. Condition section of some sample rules

Rules	Condition Section of Rules
R_1	$DI_A > DI_B$
R_2	DI_C= {steel *OR* copper *OR* cement}
R_3	DI_B=20 *AND* $DI_D <= DI_A$
R_4	DI_A<20 *AND* DI_C= {brass}

Table 6. Domain of conditional variables of the above table's rules

0< DI_A <100	-110< DI_B <50	-2000< DI_D <50
DI_C= {shoe, steel, cement, copper, brass, carpet, orange, silver, gold}		

According to this fact that both condition section of all rules and the domain of conditional variables of all condition sections are specified before run time and on the assumption that probability distribution of conditional variables is equal and logical statements are independent, we can calculate correctness probability of each logical statement before system's run time mathematically. Assumption of equal probability distribution of conditional variables means that occurrence probability of all valid values in the domain of those variables is equal.

Here we are going to calculate correctness probability of the condition of rule R_1 according to assumptions and known information mentioned in tables (5), and (6). As it is shown in figure (2), DI_A is greater than DI_B in each point inside the trapezoid S1 and DI_A is smaller than DI_B in each point inside the triangle S2. So correctness probability of $DI_A > DI_B$, on the assumption that area of the trapezoid S1 is equal to S_{S1} and area of the triangle S2 is equal to S_{S2}, is calculated according to relation (4):

$$P(R_1) = \frac{S_{S1}}{S_{S1} + S_{S2}} \quad (4)$$

Then the appropriate values are assigned to the variables of relation (4), so we will have:

$$P(R_1) = \frac{(110*100) + ((100+50)*25)}{100*160} = \frac{14750}{16000} = 0.92$$

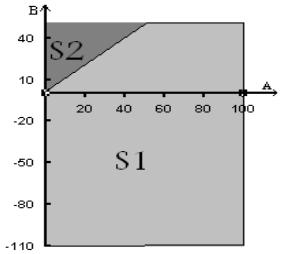

Figure 2. Calculation the correctness probability of the condition of rule R1

In this same manner, correctness probability of all rules (execution probability of the action section of all rules) is calculable before system's run time. After calculating the execution probability of action section of rules, we can calculate execution time of rules according to relation (2). Now system starts its work and scheduler, in each instant, select the rule which has the shortest execution time through activated rules with true condition to pass the operations of its action section to transaction execution unit for execution. But as we mentioned before, in this method, probability distribution of conditional variables is assumed equal in calculating of the rules execution time. But in real systems, probability distribution of conditional variables isn't often equal. So we act as below to correct this assumption to calculate execution time of rules more exact during executing the system.

After that the system starts to work, in specific intervals (δt_i), for every conditional variable of the condition section of the rules, all taken values with their taken intervals are recorded. Table (7) shows the above issue for DI_A.

At the end of every δt, occurrence probability of every value of the domain of every conditional variable is corrected based on taken interval by relevant variable during last δt [1]. This process is repeated for each value of the domain of every conditional variable until the changes rate of occurrence probability of that value reach the agreeable extent (e.g., 0.001 and in general mode ε). At this time, new value is replaced with primary default value calculated based on the assumption of equal probability distribution. It is evident that the smaller value ε, the more exact calculation of distribution probability of the conditional variables that finally leads to more exact calculation of the rules execution time. When occurrence probability of all domain's values of all conditional variables existed in the condition part of a rule such as rule R are updated, correctness probability of the condition of rule R is mathematically calculated

Table 7. Manner of recording the taken values with their taken intervals for DI_A by passing the time

Specific Intervals / Conditional Variable	δt_1 (1000 min)		δt_2 (1000 min)		…	δt_n (1000 min)	
	taken values	taken intervals (by min)	taken values	taken intervals (by min)		taken values	taken intervals (by min)
DI_A	2	10	23	8		20	102
	67	5	47	28		71	50
	⋮	⋮	⋮	⋮		⋮	⋮
	32	39	86	95		11	320
	14	15	4	5		1	55

again (updated) based on the real probability distributions of its conditional variables and the new value is replaced with primary value. Ultimately execution time of rule R is updated based on relation (2), when correctness probability of its condition section and the execution time of all its children are updated. So, after passing a short time from start of executing the system (which this time is insignificant in compare with total executing time of system), execution time of all rules are updated with a satisfied precision (which amount of this precision is depend on value ε). We do all these calculations when the system is idle. In other words, we determine the length of δt in such a manner that its termination and idle times of system get concurrent.

After implementation of new approach in the framework mentioned in section 3, we evaluate its operation. Table (8) shows the percentage of optimizing the rule scheduling in E_x-SJF_{PRO}-V.2.8 in compare with E_x-SJF_{PRO}-V.1.8 (the most effective existing rule scheduling approach), based on three evaluated parameters: Average Response Time, Response Time Standard Deviation, and Throughput.

Table 8. Percentage of rules scheduling improvement in E_x-SJF_{PRO}-V.2.8 in compare with E_x-SJF_{PRO}-V.1.8

Evaluation Criteria / Mode	Average Response Time	Response Time Standard Deviation	Throughput
Immediate	9%	13.6%	11%
Deferred	15.8%	31.9%	15.8%
Composite	17.6%	16.4%	26%

Results of experiments show that by adding a new estimation module, execution time of rules are calculated more exactly (the amount of this precision depends on value ε) and leads to improving the Average Response Time, Response Time Variance and Throughput of ADBS. The new algorithm dose not impose any overhead on the ADBS. So the Time Overhead per Transaction and CPU Utilization of E_x-SJF_{PRO}-V.2.8 and E_x-SJF_{PRO}-V.1.8 are equal.

4. Conclusions

In this paper, we first defined Active Database System and rules processing cycle. Then we expressed the position and importance of rules scheduling process. Afterwards existing rule scheduling approaches were introduced. Then we showed the results of comparison and evaluation of these approaches which had already been obtained by using a defined framework based on five evaluation criteria. Then, to improve the most effective existing rule scheduling approach, we developed a new algorithm for estimation of rule execution probability and developed a new rule scheduling approach based on this algorithm. The new approach was called E_X-SJF_{PRO}-V.2.8. Then we compare and evaluate E_X-SJF_{PRO}-V.2.8 and the most effective existing rule scheduling approach i.e. E_X-SJF_{PRO}-V.1.8 in the mentioned framework with each other. Results of experiments show that E_X-SJF_{PRO}-V.2.8 has the positive impact on Average Response Time, Response Time Standard Deviation, and Throughput of ADBS and doesn't have any negative impact on Time Overhead per Transaction and CPU Utilization.

5. References

[1] Rasoolzadegan, A.; Abdollahzadeh; "ADSS: Active Database System Simulator to Compare and Evaluate Rule Scheduling Methods", Technical Report-CE/TR.RP/85/01, Intelligent Systems Laboratory, IT & Computer Engineering Department, Amirkabir University of Technology (Tehran Polytechnic), Tehran, 2006.

[2] Vadua, A.; Rule Development for active database, PhD Thesis, CS Department, University of Zurich, 1999.

[3] Sivasankaran, R. M.; Stankovic, J. A.; Towsley, D.; Purimetla, B.; Ramamritham, K.; "Priority Assignment in Real-Time Active Databases", The International Journal on Very Large Data Bases, Vol. 5, No. 1, January 1996.

[4] Ceri, S.; Gennaro, C.; Paraboschi, S.; Serazzi, G.; "Effective Scheduling of Detached Rules in Active

Database", IEEE Transaction on Knowledge and Data Engineering, Vol. 15, No.1, 2003.

[5] Rasoolzadegan, A.; Alesheykh, R.; Abdollahzadeh, A.; "A New Approach for Event Triggering Probability Estimation in Active Database Systems to Rule Scheduling Improvement", 2nd IEEE International Conference on Information & Communication Technologies: From Theory To Applications, Damascus, Syria , April 24 - 28, 2006.

[6] Rasoolzadegan, A.; Alesheykh, R.; Abdollahzadeh, A.; "Measuring Evaluation Parameters in Benchmarking Rule Scheduling Methods in Active Database Systems", The IEEE International Conference on Computer and Communication Engineering, Kuala Lumpur, Malaysia, 2006.