3rd International Conference on Knowledge-Based Engineering and Innovation (KBEI-2016)
Dec. 30th, 2016
(Tehran Provine Payame Noor University) – Tehran, Iran

A New Logistic Algorithm Contrasts Transactions Deadlock By Using Mamdani Controller

Elmira Janani
Dept. of Computer and Informatics Engineering
Student Of Payam-e-Noor University
Tehran, Iran
Elmira.janani@yahoo.com
Tel Number: +989120138927

Mehdi Khalili
Dept. of Computer and Informatics Engineering
Professor Of Payam-e-Noor University
Tehran, Iran
Khalili.mehdi@yahoo.com
Tel Number: +989123628311

Abstract— **Today, algorithms are being used in order to solve deadlocks through prevention or detection and elimination after their occurrence. Therefore, logistic algorithm presented in this article to solve deadlocks' issues.In contrast to the current algorithms, it faces to this problem in two sections by using fuzzy controllers.So, first it reduces the possibility of deadlock occurrence and then detects and removes the occured deadlock. Suggested algorithm follows fuzzy and Aristotelian logic so, some logical rules are defined in each section. Then, an algorithm that uses of Mamdani controllers has been applied to control the deadlock. Simulation and experiments reveal that the proposed algorithm has three major advantages: first, its execution cost in system is less than current algorithms. Second, depending on our expectation and the type of system's operation, it is so flexible and adjustable. And third, it is more efficient than existing algorithms.**

Keywords—transaction, deadlock, preventing and detecting deadlock algorithms, fuzzy logic, Mamdani controller

I. INTRODUCTION

Deadlock is an important issue which has to be paid attention in parallel execution of transactions.Deadlock occures when two or more transactions require resources owned by other transactions to complete their execution periodically[1].Transactions deadlock have four characteristics: mutual exclusion, hold and wait, no preemption and circular waiting[2]. A deadlock can be defined exactly by an oriented graph called system resource allocation graph. This graph includes a set of vertices V and a set of edges E[3]. Generally, algorithms which have been defined to solve the deadlocks, dividing in two categories: algorithms that prevent deadlocks and algorithms that detect and resolve them[4].Two-partly algorithm introduced in this article by using fuzzy concepts, which first decreases the possibility occurance of deadlock and then occured deadlocks detected and some involved transactions will be introduced to be placed in rollback state.

II. REVIEW OF RELATED LITERATURES AND STUDIES

Most of the algorithms that have been presented to deal with transactions deadlock so far are a set of algorithms based on some relatively complicated and unflexible rules[5].We can use algorithms avoiding deadlocks if we want the system never enters a deadlock state. On the other hand, there is a possibility of deadlock occurance in another algorithm but using the discovery and resolved-deadlock algorithm, occured deadlocks detected and roll backed[6]. Eventually, both of these algorithms either prevent of running many transactions or rolling them back. When the possibility of occurring deadlock is high, avoid deadlock algorithm has been used but if the probability of creating a deadlock is low, the discovery and resolved-deadlock being applied[7].

A. Deadlock prevention algorithms

Generally, there are two approaches about preventing deadlocks: single transaction deadlock request fulfilment approach, all deadlock request fulfilment approach[8].
The first approach implies that among transactions and by meeting a transaction lock's demand, circular waiting will not be created. In second approach, all transactions have to ask their required locks at once, so in this way, a transaction will gain all the required locks to run or no locks allocated to it. As a result, a deadlock won't be created. Implementing this algorithm requires highlighting all the necessary locks before a transaction begun because in this method, all the locks allocated to a transaction in a phase or the transaction stayed waiting. There are two main disadvantages in this algorithm:
First, detecting and preventing items that a transaction will require are difficult and costly, and second, utilizing various data items decrease strongly because data items can stay in lock mode without being used for a long time[9].

B. *Discovery and Resolved-Deadlock Algorithms*

In this algorithm, discovery deadlock algorithm performed at specific timescales and in the condition of detecting a deadlock, resolved-deadlock algorithm implemented[9]. Also a graph formed in order to detect a deadlock based on a transaction's resources and its required one and in the condition of being a loop in that graph, detected deadlock and required actions for rolling back those transactions involved in the deadlock, have been done. Timescales that the deadlock discovery algorithm runs, depends on the number of times that a deadlock occurs and the number of transactions affected by deadlocks as well[10].

III. Fuzzy Controllers suggestion to Solve Deadlock Problem

Fuzzy logic sees the world the way it is, not absolutely dark or white, but gray[11]. Fuzzy logic, which is also usable in various branches of artificial intelligence, its logic is similar to humans'[12]. Unlike classical logic that knows everything completely right or wrong, fuzzy logic use approximate words and phrases just like humans to describe issues[13]. The fuzzy logic controller systems aim at controlling complicated processes through human experiences[14]. The current controlling systems made by physical models and need a strong engineering background. Since creating a model is reality abstraction process, therefore, always a model is the simplified version of a real process. So many issues would not be considered in modeling such as noises which can create disorders and problems in system's future, while humans are able to control that system without owning a model[15]. Fuzzy controller tries to apply humans' algorithm. One of these controllers is Mamdani controller which the proposed algorithm in this article inspired from this controller[16].

IV. Mamdani Controller

Mamdani controller's main idea is describing a condition through linguistic variables and applying them as an input of controlling rule[17]. Mamdani controller forms in four stages:

A. *Fuzzification*

the first stage to determine the in-put's degree of membership to each relevant fuzzy set[18].

B. *Rules evaluation*

in this level, the fuzzified inputs firstly applied. If the given fuzzy law has multiple predicates, AND or OR used to calculate a relationship between the predicates of rule. As in (1), OR used to assess the detachable leadings of the fuzzy rules[19].

$$\mu_{A\cup B}(x) = \max\{\mu_A(x), \mu_B(x)\} \qquad (1)$$

Similarly,as in (2) to calculate rule's leading relation, AND is being used[19].

$$\mu_{A\cap B}(x) = \min\{\mu_A(x), \mu_B(x)\} \qquad (2)$$

In Equation (1) and (2), $\mu_A(x)$ and $\mu_B(x)$ are leadings of the fuzzy rules.

C. *Calculating Resultant*

The membership functions for all cut results from the previous step are combined into a single fuzzy set in this step[18].

D. *Defuzzification*

As in (3),through gravity center formula, the point that shows the gravity center of result fuzzy set has been calculated[18]:

$$u = \frac{\sum_i \mu(x_i) x_i}{\sum_i \mu(x_i)} \qquad (3)$$

In Equation(3), X_i is values in fuzzy sets and $\mu_{(X_i)}$ is their degree of occupation.

V. Suggested Algorithm

The suggested fuzzy algorithm formed in two parts: (1) first part which decreases the possibility of creating a deadlock, (2) second part which detects and remove the occured deadlock. As it has been mentioned in their definitions, there is no certainty in deadlock issues and just they include a special kind of a fuzzy concept. The algorithm proposed in this article to solve the deadlock problem is combination of fuzzy and Aristotelian logic and the main idea is from Mamdani controllers. We present the following rule in this article: if the number of free system resources is low and the number of active transactions is high, the new transaction must wait. Although this rule seems simple, but if it is implemented properly in a system, then it will avoid the possibility of deadlock occurance. It must be noted that in the condition of low number of free resources system and high number of active transactions, and also all the new transaction given the possibility of being active, deadlock potential will be increased. The amount of system's open resources and active transactions is quite clear and accessible at any moment and by keeping two counters, they can be implemented.

Based on the fuzzy concepts of low and high mentioned in the rule, fuzzy logic and Mamdani controller have been used in the suggested algorithm. This rule used in the 6 following phases:

A. *Phase no.1*

The first phase of algorithm is fuzzificating the system input parameters. In this case, input parameters are free system resources and the number of system active transactions. An algorithm suggested by Dr. Mamdani is use of drawing chart[19]. Data block is the total system resources. "Fig.1" can be drawn based on our need and perception of free resources.

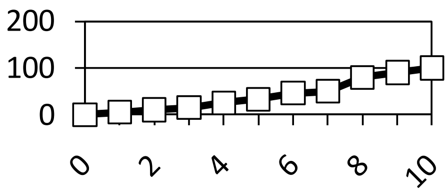

Fig.1.Degree of being busy in terms of system resources

As in "Fig.1", horizontal axis represents the system resources, and vertical axis shows degree of being busy. By defining the chart definition, degree of occupation can be calculated in certain amount of resources.

In order to simplify and decrease calculation's volume without having much impact on the operation and instead of drawing the curve, we divided the horizontal axis of the graph into appropriate intervals in the suggested algorithm and specified degrees at each extremum and matched them with lines. Dividing resources into 10 intervals is a strategy used in this algorithm that is leading to the desirable flexibility.

Assume that a graph of ordered couples (xi, yi). If horizontal axis divided into 10 intervals, we will have 11 pairs (x_i,y_i) that x_i will be on the upper level of i-th period. As in (4),in order to increase speed, the coefficient angle line calculated where xi represents the extremums of intervals and yi shows their degree:

$$a_i = \frac{y_i - y_{i-1}}{x_i - x_{i-1}} \qquad (4)$$

It is enough to have three 10-element linear arrays, where the first array holds all a_i-s, the second contains all x_i-s and the third has all y_i-s.As in (5),we show system resources with x and want to calculate its degree of occupation, indexed with Y. In second array, an interval including X can be determined through the following formula, by doing four comparison.

$$y = a_i \times (x - x_i) + y_i \qquad (5)$$

Through the same trend,as in "Fig.2" and "Fig.3", we can highlight that lot of active transactions system Degree and degree of expected transactions from their specific arrays.

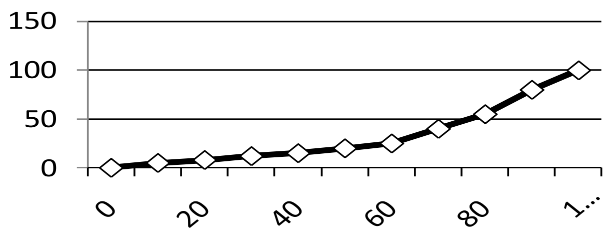

Fig.2. The high degree of active transactions system

In "Fig.2", horizontal axis represents active transactions, and vertical axis shows the high degree of it.

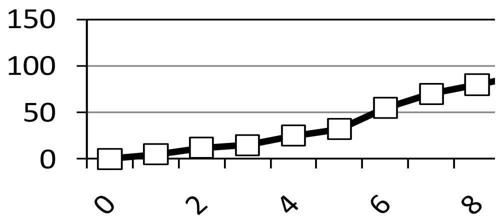

Fig.3.Degree of transactions expectation

In "Fig.3", horizontal axis represents system transactions, and vertical axis shows the degree of transactions expectation.

B. *Phase no.2*

In this phase, we calculate the minimum degree of conditions for each rules. If we use those signed used in fuzzy logic and demonstrate free resources with FR and active transactions with AT, as in (6), μ_{free}(FR) represents the degree of free resources and μ_{high}(AT) represents large degree of active transactions and μ_c shows a rule's degree of initial conditions. Based on this note, the total calculation of the second phase is as follow:

$$\mu_c = Min(\mu_{free}(FR), \mu_{high}(AT)) \qquad (6)$$

If there are multiple rules, it has to be calculated for each ones.

C. *Phase no.3*

In 3rd phase, the minimum of (μ_{ci},μ_{rfxi}) is calculated which represents degree of operation that must be acted in second side of i rule. In order to calculate μ_{rfxi} we can use the mentioned algorithm in phase no.1 just with a difference that we have to specify X axis.

$$\mu_{R_i} = Min\left(\mu_{C_i}, \mu_{wait}\right) \qquad (7)$$

As in (7), μ_{Ri} shows degree of output operation of i rule. To calculate it an 11-element array is required. The values that fill the i-th elements in this array are the minimum of the μ_{ci} and the i-th element of the μ_{rfx} array.

D. Phase no.4

If we increase the number of rules, we must do previous actions for each one and then we integrate arrays from previous stage together. The algorithm of these arrays' integration has been so simple. It just needs to replace maximum i-th elements resulted in 3rd phase in the i-th elements of the array. Because we use a rule in this algorithm, the output is the same as previous phase and it is not necessary to do an action in this phase.

E. Phase no.5

To calculate the final degree of operation from the following formula, first the weighted average of the diagram resulted from previous stage is calculated.

$$\bar{p} = \frac{\sum_{i-0}^{10} \mu_i p_i}{\sum_{i=0}^{10} \mu_i} \qquad (8)$$

As in (8), pi is the same numbers obtained in the horizontal axis in 3rd phase which is pi=i here.µi is the same number saved in arrays obtained from phase no.4.The output of the formula is a number between 0-10 that by implementing an action against the one in the first phase, the degree of final operation will be found out.Wait defined in this final operation.

F. Phase no.6

According to the obtained degree in previous stage, we decide if Wait done or let the transaction be active. Choosing the Wait threshold can be adjusted depending on the system operation. In normal mode, the default threshold is 50 which means if the output degree is over 50, Wait will be done and in the condition of being bellow 50, transaction will be active and placed in Schedualer.

As in "Fig.4" and "Fig.5", the following rule will be applied in second section: If a transaction stays pending for a long time to obtain a lock on a resource and the number of active transactions are low and it also owns few resources, transaction should be rolled back.

If the transaction does not meet its request, it is most probable that it has been placed in a periodic ring-term demand which is a sign of a deadlock. The third condition says that, greater transactions or those that have been run for a long time are in priority and in the same condition, younger and smaller transactions would be rolled back.

This rule is able to discover the created deadlocks and introduce transactions participated in them for rolling back. Here are the second rule's charts that are suggested and applied in this article:

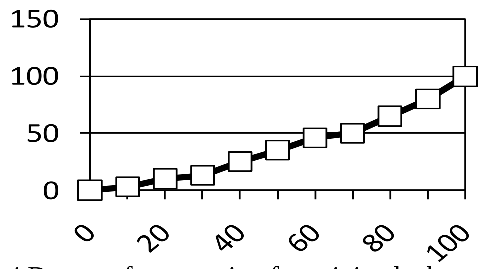

Fig.4.Degree of expectation for gaining lock

In "Fig.4", horizontal axis represents active transactions system, and vertical axis shows the degree of expectation for obtaining lock.

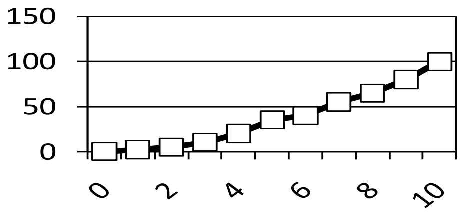

Fig.5.Degree of rolling back transaction probability

In "Fig.5", horizontal axis represents active transactions system, and vertical axis shows the degree of rolling back transaction probability.

VI. Comparing the Result of Suggested Fuzzy Algorithm with the Result of Implementing Prevention Algorithm & Discovery and Resolved-Deadlock on the Same Data

As in "Fig6" and "Fig7", two scales including clock and rollback have been considered to determine optimized algorithm[20].In this regard, Prevention Algorithm, Suggested Fuzzy Algorithm, and also Discovery and Resolved-Deadlock carried out on the same input data.As you see, the number of Suggested Fuzzy Algorithm's clock and rollBack are less than two other algorithms. So it operated more efficient in the same input data.

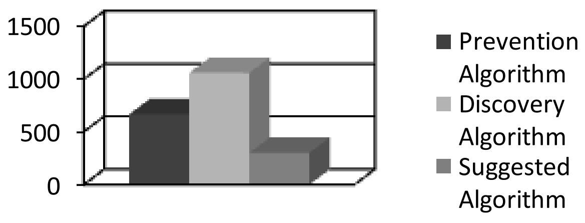

Fig.6.Number of clock per each of algorithms

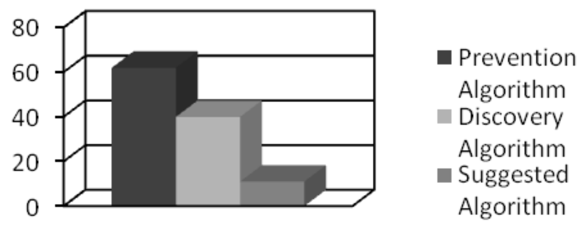

Fig.7.Number of Rollback per each of algorithms

VII. CONCLUSION

Through investigating 3 algorithms, Prevention, Suggested Fuzzy, and also Discovery and Resolved-Deadlock, it is clear that prevention deadlock algorithm requires fundamental information from transaction structure and their requested resources but in most cases, this information is not available[21]. Discovery and Resolved-Deadlock protocols mostly use complicated structures (e.g, Wait-for graph) that the cost of processing and keeping such structures is so costly for a system in comparison with an array of structures[22].

Suggested algorithm in this article is able to solve the deadlock problems by applying simple structure and processing. First, it decreases the possibility of making a deadlock and then discovers and removes the created deadlock. Also some involved transactions will be introduced to be rolled back. The simulations and experiments reveal that not only the suggested algorithm costs less than current ones but also they are so much flexible and adjustable based on our expectation and the type of system's operation.

References

[1] Sailen Dutta Kalita, Minakshi Kalita, Sonia Sarmah, "A Survey on Distributed Deadlock Detection Algorithm and its performance Evolution", IJISET - International Journal of Innovative Science Engineering & Technology, vol. 2, no. 4, April 2015.

[2] R.C. Holt, "Some deadlock properties of computer systems", ACM Computing Surveys, pp. 179-195, Sept. 1972.

[3] Qinqin Ni, Weizhen Sun, Sen Ma, "Deadlock Detection Based on Resource Allocation Graph", Fifth International Conference on Information Assurance and Security, vol. 2, pp. 135-138, 2009.

[4] Gupta Dhirai, V.K Gupta, "Approaches for Deadlock Detection and Deadlock Prevention for Distributed systems", Research Journal of Recent Sciences, vol. 1, pp. 422-425, 2012.

[5] Brian M. Johnston, Ramesh Dutt Javagal, Ajoy Kumar Datta, Sukumar ghosh, A Distributed Algorithm for Resource Deadlock, pp. 252-256, 2011, Department of Computer Science, IEEE.

[6] O. T. Baruwa, M. A. Piera, A. Guasch, "Deadlock-Free Scheduling Method For Flexible Manufacturing Systems Based On Timed Colored Petri Nets And Anytime Heuristic Search", IEEE Transactions on Systems Man and Cybernetics: Systems, no. 99, pp. 1-16, 2014.

[7] Priyadarshini, S. S. Sane, Rutuja Jadhav, "Deadlock Detection in Distributed Database", International Journal of Emerging Trends & Technology in Computer Science (IJETTCS), vol. 2, no. 2, pp. 2278-6856, March 2013.

[8] B.M. Monjurul Alom, Frans Henskens, Michael Hannaford, "Optimization of Detected Deadlock Views of Distributed Database", International Conference on Data Storage and Data Engineering, pp. 44-48, 2010.

[9] Md. Abdur Razzaque, Md. Mamun-Or-Rashid, Choong Seon Hong, "Multi-cycle Deadlock Detection and Recovery Algorithm for Distributed Systems", Conference Paper, pp. 554-565, January 2007.

[10] Xiuyan Zhang and Murat Uzam."Transition-based deadlock control policy using reachability graph for flexible manufacturing systems".Advances in Mechanical Engineering 2016.

[11] Reza Dwi Utomo,Sumardi, Eko Didik Widianto, "Control system of train speed based on fuzzy logic controller." IEEE Trans. 24 March 2016.

[12] H. Dong, L. Li, B. Ning, Z. Hou, "Fuzzy tuning of ATO system in train speed control with multiple working conditions", Proc. of 29th Chinese Control Conf., pp. 1697-1700, July 2010.

[13] J.M. Mendel, "Uncertain Rule-Based Fuzzy Logic Systems", Prentice Hall, New Jersey, 2011, ISBN 0-13-040969-3.

[14] Patricia Ochoa, Oscar Castillo, José Soria,"Fuzzy differential evolution method with dynamic parameter adaptation using type-2 fuzzy logic". IEEE Trans,10 November 2016.

[15] Wenhui Liu, Cheng-Chew Lim, Peng Shi," Backstepping fuzzy adaptive control for a class of quantized nonlinear systems". IEEE Trans, 05 August 2016.

[16] A. Al-Odienat and A. Al-Lawama, "The advantages of PID fuzzy controllers over the conven-tional types", American Journal of Applied Sciences 5(6) (2008), 653-658.

[17] S. Mohagheghi, G. K. Venayagamoorthy, R. G. Harley, "An adaptive Mamdani fuzzy logic based controller for a static compensator in a multimachine power system", Proc. IEEE ISAP, pp. 228-233, 2015-Nov.

[18] Esmaeil Valipour Arekhloo, Saeid Pashazadeh, Seyed Naser Razavi," Modelling of Mamdani Fuzzy Inference Engine using hierarchical colored Petri nets",IEEE 02 December 2013.

[19] V. Geetha, N. Sreenath, Anil Kumar, Reddv M. Saravana, S Sundar, S Shanmuga Privan, "Deadlock Elimination of AND Model Requests in Distributed Systems", IEEE-International Conference on Recent Trends in Information Technology ICRTIT 2011, June 3–5, 2011.

[20] Yan Cai, W.K. Chan, "Magiclock: Scalable Detection of Potential Deadlocks in Large-Scale Multithreaded Programs", IEEE Transaction on software engineering, vol. 40, no. 3, march 2014.

[21] Cormac Flanagan, Stephen N. Freund, "FastTrack: Efficient And Precise Dynamic Race Detection", Proc. ACM SIGPLAN Conf. Programming Language Design and Implementation (PLDI), pp. 121-133, 2009.

[22] Deepak Kumar, Pankaj Verma," Comparative study of Mamdani & Takagi-Sugeno models for spectrum access in cognitive radio networks", IEEE,21 March 2016.