International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

COPY RIGHT

ELSEVIER SSRN

IJIEMR Transactions, online available on 27th Dec2020. Link :http://www.ijiemr.org/downloads.php?vol=Volume-09&issue=ISSUE-12

DOI: 10.48047/IJIEMR/V09/I12/113

Title: **Soft Computing Impacts in Resolving Deadlock Issues.**
Volume 09, Issue 12, Pages: 684-690
Paper Authors

Mr. Mohammed Bakhtawar Ahmed, Mr. Debojit Das

USE THIS BARCODE TO ACCESS YOUR ONLINE PAPER

To Secure Your Paper As Per UGC Guidelines We Are Providing A Electronic Bar Code

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

Soft Computing Impacts in Resolving Deadlock Issues

Mr. Mohammed Bakhtawar Ahmed
Assistant Professor, Amity University Chhattisgarh, Raipur
bakhtawar229@gmail.com
Mr. Debojit Das
Assistant Professor, Vivekanand Mahavidyalaya, Raipur
debojit.das2707@gmail.com

Abstract—Distributed nature of transactions arising at different sites and needing resources from diverse locations pose various operational problems, such as deadlocks, concurrency and data recovery. A deadlock may occur when a transaction enters into wait state that request resource from other blocked transactions. Deadlock detection and resolving is very difficult in a distributed database system because no controller has complete and current information about the system and data dependencies. In this paper, an enhanced technique for deadlock resolution is presented, which minimizes the abortion or waiting of the selected victim transactions.

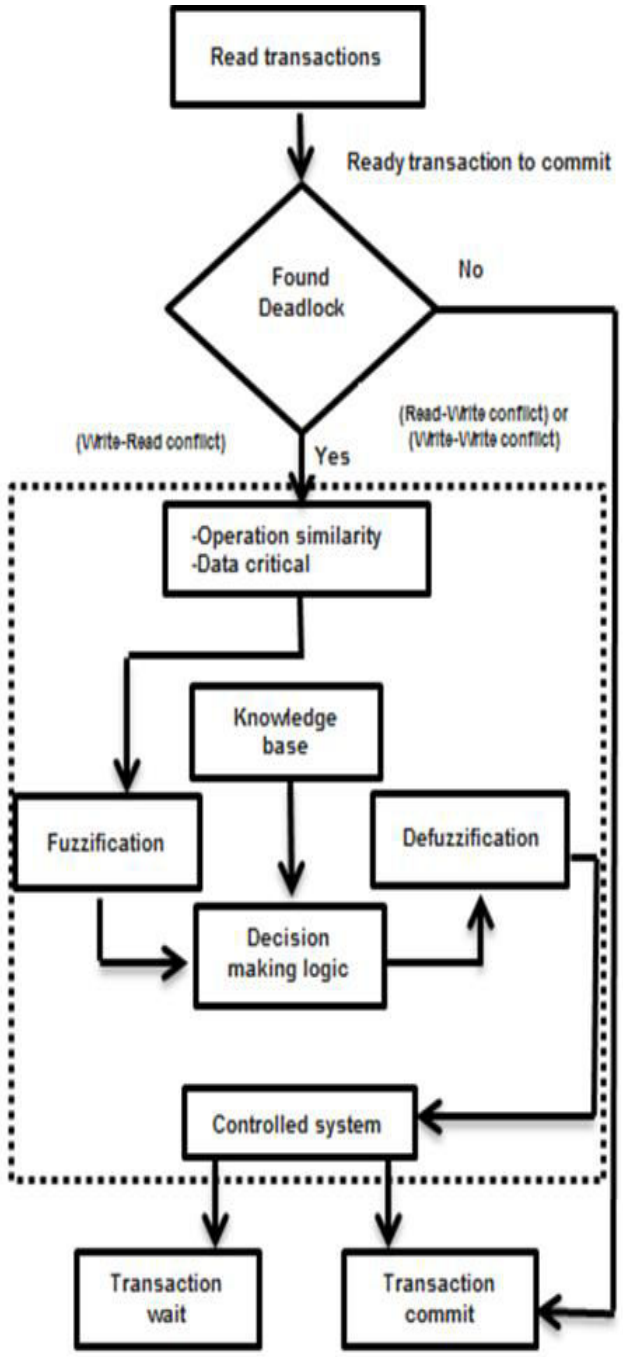
I. INTRODUCTION

Deadlock is one of the most serious difficulties in multitasking concurrent programming systems. The deadlock problem becomes further complicated when the underlying system is distributed and when tasks have timing limitations. Deadlock is a system state in which every process in some group requests resources from other processes in the group, and then waits indefinitely for these requests to be satisfied [1]. Deadlock is an undesirable situation; some of the consequences of deadlock are: throughput of the system is affected; utilization of the involved resources decreases to
zero; deadlock increases with deadlock persistence time; and deadlock cycles do not terminate by themselves until properly detected and resolved [2], [3]. The deadlock problem is intrinsic to distributed database system which employs locking as its concurrency control algorithm. Concurrency control and deadlock handling are the most important problems that must have a powerful attention when sharing information in distributed systems [1].

Deadlock resolution requires at least one of the transactions causing the deadlock to release locks. This involves a partial rollback, lock de-escalation, or most commonly a transaction termination [3]. The throughput of the entire database system depends on the efficiency and accuracy of the deadlock detection and resolution algorithms. The correctness of a deadlock algorithm depends on two conditions. First, every deadlock must be detected eventually. This constitutes the basic progress property in which any solution must have. Second, if a deadlock is detected, it must indeed exist (safety property) [4]. Incorrectly detected deadlocks due to message delays and out-of-date wait-for-graphs (WFGs) have been termed phantom deadlocks. The main disadvantage of deadlock detection schemes is the additional overhead incurred due to detection of cycles in the graph and abortion and restart of transactions upon detection of deadlocks. The distributed detection strategies may have additional overhead due to the inter site message transfers. Selection of the transaction to be aborted adds to the complexity of the scheme.

There are four techniques regularly engaged to deal with deadlocks in database systems: ignore the problem, deadlock

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org



detection, deadlock prevention and deadlock avoidance. Ignoring deadlocks is the easiest scheme to implement. Deadlock detection attempts to locate and resolve deadlocks.

Deadlock avoidance describes techniques that attempt to determine if a deadlock will occur at the time a resource is requested and reacts to the request in a manner that avoids the deadlock. Deadlock prevention is the structuring of a system in such a manner that one of the necessary conditions for deadlock cannot occur. Each solution category is suited to a specific type of environment and has advantages and disadvantages, see [4], [5] for more details.

In general, database deadlock resolution involves the following nontrivial steps [6]-[9]:

1) Select a victim (the transaction to be aborted) for the optimal resolution of a deadlock (this step may be computationally tedious).

2) Abort the victim, release all the resources held by it, restore all the released resources to their previous states, and grant the released resources to deadlocked processes.

3) Delete all the deadlock detection information concerning the victim at all sites. Execution of the second step is complicated in environment where a process can simultaneously wait for multiple resources because the allocation of a released resource to another process can cause a deadlock. The third step is even more critical because if the information about the victim is not deleted quickly and properly, it may be counted in several other (false) cycles, causing detection of false deadlocks. To be safe, during the execution of the second and third steps, the deadlock detection process (atleast in potential deadlocks that include the victim) must be halted to avoid detection of false deadlocks.

Built on top of the work suggested in [12], a similar type of approach is adopted here to resolve deadlocks based on fuzzification of the transaction's attributes to build a new rules-based priority for conflict resolution between transactions. Design of fuzzy logic or rule based non-linear controller is easier since its control function is described by using fuzzy sets and if-then predefined rules rather than cumbersome mathematical equations or larger look-up tables; it will greatly reduce the development cost and time and needs less data storage in the form of membership functions and rules. The properties of this solution are locality of transactions, and asynchronous operation. We elaborate our simulation results and justify performance gain of the proposed scheme for achieving deadlock management in database environments by

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

eliminating limi tations of the existing schemes, increasing commit rate and decrease in re-execution rate of the transactions.

III. DEADLOCK RESOLUTION WITH FUZZY LOGIC

In order to overcome shortcomings of the deadlock resolution methods discussed above to certain contain, by using transaction's features, a robust resolving scheme using both transaction's features-based and fuzzy logic is proposed as shown in Fig. 1. The suggested system utilizes fuzzy logic technique by creating a set of fuzzy rules that will form the fuzzy logic engine in order to deal with the criticalness and the similarity attributes of transactions. By using these rules, fuzzy logic will try to provide an easy conflict resolution method between transactions. The algorithm attempts to outperform the previous methods by reducing

the number of transaction waiting and increasing the concurrency level while maintaining the data valid as much as possible. Table I shows the different terms and parameters applied in the proposed system.

TABLE I: LIST OF PARAMETERS USED IN THE PROPOSED APPROACH

Parameter	Definition
D	Transaction's data object
T	Transaction
t	Current time
T_a	Active conflicting transaction
T_v	Validating conflicting transaction
RS	Data read set (The adjustment of timestamp of an active transaction iterates through the read set)
WS	Data write set (The adjustment of timestamp of an active transaction iterates through the read set)
$S(T)$	Start time of transaction T
$A(T)$	Arrival time of T
$C_t(T)$	The estimated completion time of T at time t
$E_t(T)$	The estimated remaining execution time of T at time t
$D(T)$	Deadline of transaction T
$T(Op_i, D)$	Transaction 's operation sharing with D
$f(x;a,b,c,d)$	Trapezoidal membership function

A. Resolution by using Timestamp

One of the most commonly used technique for deadlock resolution is timestamp based approach for selecting the victim. In this approach, a timestamp is allocated to each process as soon as it enters the system. The timestamp of the younger process is greater than the timestamp of older process. According to this approach, the victim is selected on this timestamps, the process with the higher timestamp is aborted, that is the youngest process is selected as the victim and is aborted in order to break the deadlock cycle. The goal behind choosing the youngest process as victim is that the youngest process would have used less resources and less CPU time as compared to older process. One problem with this technique is that it can cause starvation problem because every time a younger process is aborted which can starve the younger process from completion.

B.Resolution by using Burst time

Another approach for selecting a victim to break deadlock cycle is considering the burst time of each process. Burst time means the CPU time needed by any process for its execution. This can also be considered as one parameter for selecting a victim. The process with maximum burst time can be aborted in order to break cycle. The problem with this technique is that it can abort the process with high burst time which has been in the system for very longi.e. an older process with high burst time can be aborted which is inefficient approach.

C.Resolution by Degree

In a wait-for-graph for any system, the degree of any vertex denoting a process determines how many resources a process is holding and how many resources a process is requesting. There are two types of degrees in a directed WFG.

1.In-degree: In-degree means the number of edges coming to any node of WFG and it denotes number of request for resources held by a process.

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

2.Out-degree: Out-degree means the number of edges going out of a node in WFG denoting number of request for resources done by the node.In resolution by degree, degree of each process is calculated and process having highest degree is aborted. Degree of any process can be calculated by taking sum of in-degree and out-degree.

D.Resolution by combination of Timestamp and Burst time

Another approach for selecting victim for deadlock is using both timestamp and burst time in combination. Select a process as victim which is younger and has high burst time for resolving deadlock. The advantage with this approach is younger process which will take maximum execution time will be aborted to allow processes with less execution time to complete first.

E.Resolution by combination of Burst time and Degree

Another combination for resolving deadlock is considering Burst time and Degree both for selecting a victim. Process with high burst time and high degree should be aborted that means a process which is having more resource request and will take high time to complete will be aborted. Although, there is still the problem of older process to be aborted but the advantage with this approach is aborting process with high burst time and high degree will release maximum resources needed for completion of other process with less execution time needed.

F.Resolution by combination of Degree and Timestamp

Taking degree and timestamp both in combination for resolving deadlock can prove to be another technique for deadlock resolution. A younger transaction with high degree will be aborted.The problem of starvation in considering only timestamp will be avoided in this case as degree of the node is also considered along with timestamp in order to select victim for resolving deadlock in the system.

G.Time Efficient Deadlock Resolution Algorithm

Deadlock is a major concern in a distributed system, since resources are shared among processes at sites distributed across a network. One of the most accepted methods of deadlock handling is detection and resolution. Both deadlock prevention and avoidance strategies are conservative solutions, whereas deadlock detection is optimistic [15]. In deadlock detection and resolution, deadlocks are allowed to occur [3][15]. Periodically, or on certain conditions, a detection algorithm is executed; if any deadlock state is found, resolution is undertaken. To resolve a detected deadlock, the system must abort one or more processes involved in the deadlock and release the resources allocated to the aborted processes. Here deadlock resolution with reusable resources is considered. In resolving a deadlock state, it is desirable to minimize the number of processes to abort to make the system deadlock-free. Concept of release set is introduced here. A release set is a set of one or more processes that can be reduced if a process is aborted and its resources are released[15]. The release set is represented by R(pi). For example release set of process P7 and P5 in figure 4 is R(P7)= {P8} and R(P5)= {P6,P7, P8}.

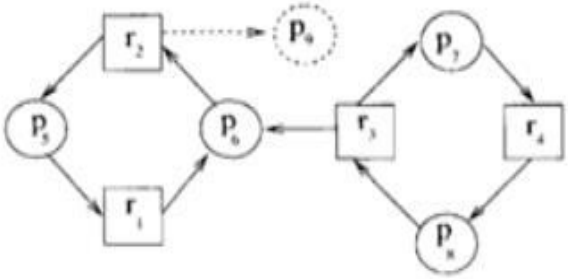


Figure 4: Deadlock cycle

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

In general, there are three phases for database's transaction: (1) Read phase: The transaction reads the values of all data items it needs from the database and stores them in local variables. In some methods updates are applied to a local copy of the data and announced to the database system by an operation named pre-write.
(2) Validation phase: The validation phase ensures that all the committed transactions have executed in a serializable fashion. For a read-only transaction, this consists of checking that the data values read are still the current values for the corresponding data items. For a transaction that has updates, the validation consists of determining whether the current transaction leaves the database in a consistent state, with serializability maintained.
(3) Write phase: This follows the successful validation phase for update transactions. During the write phase, all changes made by the transaction are permanently stored into the database [14]. the same objects Di both Tvinvalidating and active Tatransactions and at least one of the operations is a write operation, then we have a conflict (deadlock is detected). In practice, deadlock detection often assumes a simplified resource model; the system contains only reusable resources and there is only a single unit of every resource. This model makes deadlock detection simple to implement, but at the cost of detecting fewer types of deadlock.

The proposed system follows Single Request Model for static deadlock detection in which a process can have at most one outstanding request for only one unit of a resource. Since the maximum out-degree of a node in a WFG for the single resource model can be 1, the presence of a cycle in the WFG shall indicate that there is a deadlock. The rationale of choosing this request model is that it simplifies the problem of detecting the deadlock and easy to implement.
Formally, conflict can occur when
[12]: φ≠∩∈))()((TvWSTaRSDi (read-write conflict)
(1) φ≠∩∈))()((TvRSTaWSDi(write-read conflict) (2) φ≠∩∈))()((TvWSTaWSDi (write-write conflict)
(3) Here, to reflect the new developments, the attempt is to use transactions' features to solve the conflict between them through employing fuzzy controller to handle uncertainty associated with these features that affecting to the transactions' priority . temporal data items that takes into account transaction's operations such as read, write, and shared resources and criticalness that takes into account the estimated completion time of T as transaction's attribute which uses information about the importance of the transactions that will be fed into fuzzy logic engine for conflict handing. These two features were selected for ease of application and ease of calculations inside fuzzy logic engine. Suppose tmand tnare a pair of concurrent transactions, tmOpi∈, tnOpi∈, OpiandOpjoperate on the same non-critical data object D (conflicting operations). If the following condition is satisfied [12]: ≤∝−),(), (DOpjTDOpiT (4) ∝is the threshold value whose value depends on the application semantics, then Opi and Opjare said to be operation similarity, notated byOpjOpi≈. Furthermore D is critical trueDTCt=), (if:),(),(TdDTCt≤)(), (TEttDTCt+= (5) Criticalness measures how criticalit is that a transaction meets its timing constraints. Different transactions have different criticalness. Furthermore, criticalness is a different concept from deadline because a transaction may have a very tight deadline but missing it may not cause great harm to the system. Here, expected execution time is very hard to predict but can be based on estimate or experimentally measured value of worst case execution time. adaptive in nature and can also exhibit increased reliability, robustness in the face of changing transaction's features. The first step in the design of a fuzzy logic controller is to define membership functions for the inputs; three fuzzy levels or sets are chosen and defined by the following library of fuzzy-set values for the similarity (non-similar, similar, very similar) and critical attributes (very critical, critical, non-critical) of

International Journal for Innovative Engineering and Management Research
A Peer Revieved Open Access International Journal
www.ijiemr.org

transaction as shown in Fig. 2a,2b. For a given crisp input, fuzzifier finds the degree of membership in every linguistic variable. The number of fuzzy levels is not fixed and depends on the input resolution needed in an application. The larger the number of fuzzy levels, the higher is the input resolution. The fuzzy controller utilizes trapezoidal membership functions on the controller input [13]. Membership functions allow us to graphically represent a fuzzy set. The x axis represents the universe of discourse, whereas the y axis represents the degrees of membership in the [0,1] interval [19]. Simple functions are used to build membership functions. Because we are defining fuzzy concepts, using more complex functions does not add more precision. The trapezoidal membership function is chosen due to its simplicity. All of membership' parameters are numerically specified based on the experiences to handle transactions. In our case, all fuzzy levels have the same space on the number line. The trapezoidal curve is a function of a vector, x, and depends on four scalar parameters a, b, c, and d, 18International Journal of Modeling and Optimization, Vol. 5, No. 1, February 2015Step 2. Deadlock detection: When access has been made Step 3. Transaction' attributes extraction: The proposed method employs the concept of similarity for non-operations to obtain a better real-time performance, and the transaction criticalness criterion in order to favor transactions with higher importance in data conflict resolution. Furthermore, the system exploits fuzzy critical.

IV. SIMULATION RESULTS

In this section, we conducted an extensive set of simulation experiments using the above mentioned parameters in Table I through MATLAB and PHP languages. Wait percentage (Wait %) and Commit percentage (Commit %) were used as measures for the performance metrics in our simulation results. Wait % (how many transactions wait due to violation of serializability before final commit from the total number of transactions taken for concurrent execution) is the percentage of input transactions that have non critical attribute and have less than 0.6 in the similarity scale and Commit % (how many transactions successfully committed execution from the total number of transactions taken for concurrent execution) is the percentage of input transactions that have very critical attribute and have greater than 0.6 in the similarity scale (according to fuzzy system rules). We conducted simulation under normal and heavy loads with various settings of workload parameters such as number of transactions, transaction workload (simple or complex transaction) and with other corresponding parameter values.

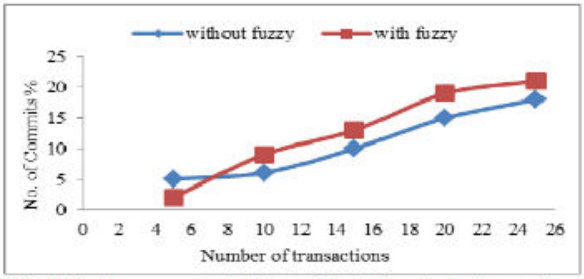


Fig. 3. Performance graph of commit rate for simple transaction.

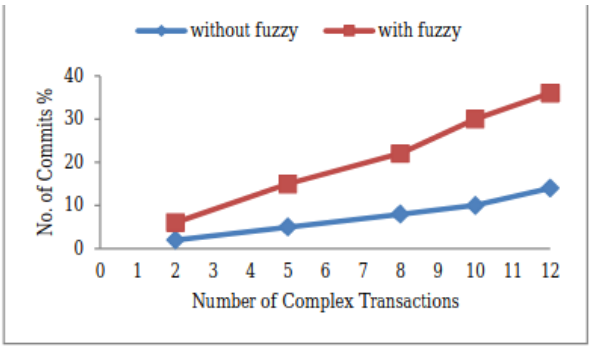


Fig. 4. Performance graph of commit rate for complex transaction.

V.CONCLUSION

Deadlock can occur in any concurrent system and is often difficult to debug. Existing deadlock resolution techniques are either impractical for large software database systems or over-simplified in their assumptions about deadlock-sensitive resources. In this paper, we propose fuzzy-based deadlock resolution, a novel database system mechanism that

dynamically handles deadlock in database applications with the capability of predicting the correctness of the transactions history in case it is rescheduled. The proposed system improves the drawbacks of the existing schemes by prioritizing the transactions based on their features. The suggested system increases the overall commit rate of the system and decreases the rate of waits. The system employs the concept of similarity between conflicting to obtain a better real-time performance, and the transaction criticalness criterion in order to favor transactions with higher importance in data conflict resolution. Furthermore, the system exploits fuzzy 20International Journal of Modeling and Optimization, Vol. 5, No. 1, February 2015Experiment 2. Comparison of waits% and commit% for Experiment 3. Comparison of wait% and Commit% under logic approach as the famous artificial intelligence technique to merge transaction's features to provide an easy conflict resolution method between transactions. The advantages of proposed scheme are 1) transactions data item access priority is maintained to ensure serializability without aborting the transactions. 2) the cost of waiting time of the transaction to execute is less than the cost of re-execution of the transaction. Hence, transaction can wait little more to acquire a data item than to access and get aborted 3) the transaction, which has done more work, is given higher priority, as it will finish early if given more privilege. Finally 4) the overall through put of the system increases by sacrificing a small amount of waiting time and overhead is conserved. Also, a simulation implementation and a performance comparison between fuzzy and non-fuzzy real-time deadlock control methods show that our method can ensure a very well real-time performance while guaranteeing temporal consistency and can even outperform non-fuzzy method in many cases. Moreover, we can try to implement our proposed method on a real-time database test platform and on a real database management system to obtain more accurate results.

REFERENCES

[1] P. Sapra, S. Kumar, and R. Rathy, "Deadlock detection and recovery in distributed databases," International Journal of Computer Applications, vol. 73, no. 1, pp. 32-36, July 2013.

[2] S. Singh and S. Tyagi, "A review of distributed deadlock detection techniques based on diffusion computation approach," International Journal of Computer Applications, vol. 48, no. 9, pp. 28-32, June 2012.

[3] S. Selvarai and R. Ramasamy, "An efficient detection and resolution of generalized deadlocks in distributed systems," International Journal of Computer Applications, vol. 1, no. 19, pp. 1-7, 2010.

[4] M. Goswami, K. Vaisla, and A. Singh, " VGS algorithm: an efficient deadlock prevention mechanism for distributed transactions using pipeline method," International Journal of Computer Applications, vol. 46, no. 22, pp. 1-9, May 2012.

[5] S. Gupta, "Deadlock detection techniques in distributed database system," International Journal of Computer Applications, vol. 74, no. 21, pp. 41-45, July 2013.

[6] F. Tanga, I. Youb, S. Yuc, C.-L. Wangd, M. Guoa, and W. Liue, " An efficient deadlock prevention approach for service oriented transaction processing," International Journal of Computers and Mathematics With Applications, vol. 63, no. 2, pp. 458-468, 2012.