Performance Analysis of Victim Selection Algorithms in Distributed Systems and Proposal of Weight Based Resolution Strategy

geetha venkat[*1], N.Sreenath[2]

[1]Department of Information Technology, Pondicherry Engineering College, India
[2]Department of Computer Science & Engineering., Pondicherry Engineering College, India
vgeetha@pec.edu

***Abstract*- Deadlocks affect the performance of all systems that support concurrent execution of transactions. Presence of deadlocks is usually detected by checking for cycles in Wait-For graph. Once deadlocks are detected, the cycle can be broken by aborting one of the transactions (Victim). Main objective of victim selection is avoiding starvation. This paper analyses the performance of various victim selection algorithms given in the literature to find out how optimal they are with respect to other desirable parameters of a system like throughput, fairness, resource utilization and resolution latency apart from starvation. This paper also proposes weight based resolution algorithm to dynamically select least cost victim.**

Keywords- Distributed System; Resolution; Transaction Attributes; Resource Characteristics, Victim Selection

I. INTRODUCTION

Deadlock detection is a reactive strategy for handling deadlocks. It is the best mechanism for systems with lower and moderate number of deadlocks. Deadlock can be usually detected by checking for presence of cycle in Wait-For Graph (WFG). Once a cycle is identified, one of the transactions should be chosen as victim. Aborting it will break the cycle and thus eliminate deadlocks. The victim thus selected needs to rollback and restart later. Hence the negative outcome of the deadlock resolution is the possibility of penalization of same transaction again and again i.e., starvation.

In distributed systems, detection of deadlock is more difficult than in centralized systems. This is because the resources are distributed in different sites and transactions access them from any of these sites. They communicate through messages only. Hence in order to know the wait-for status of the transactions, a Global WFG has to be constructed. Selection of a victim using this GWFG is complex. Zobel [1], Newton [2] and Singhal [3] have done survey on various deadlock handling techniques, but they have not focused on victim selection algorithms for deadlock resolution. A. Moon and H. Cho [4] have compared the performance of deadlock handling techniques against the attribute of throughput alone.

Hence it is proposed to analyze the performance of the existing victim selection algorithms. The paper is organized as follows. The existing victim selection algorithms are described in Chapter II and their characteristics are analysed in Chapter III. Chapter IV proposes the cost based victim selection algorithm and Chapter V concludes the paper.

II. RELATED WORKS

An Several algorithms have been proposed in the literature for the selection of victims under different criteria as given below:

A. Selection Criteria: Youngest [5]

Transaction Attribute: Arrival time or Age

Transaction that has arrived latest or whose time stamp is greater than all the participating transactions is chosen as the victim. This assumes that the later transaction would not have done much progress and hence aborts the latest transaction. It is highly fair and provides linear response time as it serves in first come FIFO basis.

B. Selection Criteria: Minimum History [5]

Transaction Attribute: History

The transaction that has been aborted least number of times so far (also called as history) will be chosen as the victim. This ensures elimination of starvation.

C. Selection Criteria: Least Priority [6]

Transaction Attribute: Static Priority

The transaction having the least static priority will be aborted. This helps to decide the order of execution, given a collection of transactions. The priority of the transactions can be statically fixed by the users or the domain.

D. Selection Criteria: Maximum Size [7]

Transaction Attribute: Size

The transaction, whose code size is the largest among all the transactions currently running, will be aborted. As the transaction size increases, it is assumed to consume more resources and finish execution much later. Hence transaction with largest size is chosen as victim. This improves the throughput of the system as more number of smaller transactions is finished in the given time.

E. Selection Criteria: Minimum number of Locks [5]

Transaction Attribute: In-degree in Wait for Graph

Transaction that has acquired the least number of

DOI: 10.5963/IJCSAI0204005

resources so far (inferred by the least number of grant messages represented by in-degree in WFG) is chosen as victim. The transaction is chosen only based on its current resource holding status and hence may improve the throughput of the system. The resource utilization improves, because of the selection of a victim which has locked minimum number of resources in the system so far, and does not penalize a transaction on any other criteria.

F. Selection Criteria: Maximum Number of Cycles [8]

Transaction Attribute: Cycle participation

Transaction involved in maximum number of deadlock cycles will be aborted. Normally, it is expected to choose one victim per cycle. By using this algorithm, the number of victims may be reduced. Gary and Johnson [9] have stated that identification of minimum number of victims at run time is NP complete. Hence number of transactions rolled back is lesser and hence throughput increases.

G. Selection Criteria: Maximum Edge Cycle [8]

Transaction Attribute: in-degree+ out -degree

This resolution is based on maximum participation of a transaction in more number of cycles. The possibility could be that the transaction already holding high priority resources and further requires more number of resources held by other transactions. Hence there is more number of edges. It is not only based on transaction attribute, but also based on resource attribute. It is the sum of request edges and grant edges.

H. Selection Criteria: Blocker [5]

Transaction Attribute: Random blocker, current blocker

The transaction that has caused the deadlock is aborted in this algorithm. The overhead of selecting a victim is nil in this case. It also reduces deadlock resolution latency. But other attributes are suboptimal.

Selection Criteria: Minimum work done [5]

I. Transaction Attribute: Resource Consumed

Transaction that has consumed least amount of resources is chosen as the victim.

Selection Criteria: Initiator

Transaction Attribute: transaction that has initiated the deadlock detection

The transaction, which had initiated the deadlock detection phase on time out, is chosen as victim. This minimizes the deadlock resolution latency in a distributed system, as initiator ID is always communicated to all sites.

J. Selection Criteria: Maximum Release Set [10]

Transaction Attribute: holding maximum number of resources

Transaction holding more number of resources which, when aborted will benefit maximum number of transactions, is chosen as victim.

K. Selection Criteria: Minimum Number of Submitted Operations [12]

Transaction Attribute: Number of submitted operations

The transaction which has done minimum work so far is chosen as the victim.

L. Selection Criteria: Low Priority + Least Resource Priority + Min. Work Done [14]

Transaction Attribute: low priority + minimum work done

This algorithm works in three phases. In the first phase a set of low priority victims are selected. In Second Phase, victims holding higher priority resources from the first phase list are chosen. In phase three, victim which has done least work done is aborted from second phase list.

M. Selection Criteria: Minimum Abortion Cost [15]

Transaction Attribute: Age and work done

Abortion cost is a function of number of currently submitted operations and transaction age and given as, Abortion cost = $\propto$ N (T) +βt (T), where $\propto+\beta = 1$ and N (t) – no of currently submitted operations, t (T) – age of transaction. $\propto$ and β are weights to choose between age and work done. Age improves fairness and work done improves throughput.

III. PERFORMANCE OF EXISTING ALGORITHMS

All Based on the definition of the above victim selection algorithms, their characteristics along with their time complexity can be summarized as in Table 1.The time complexity helps to determine the deadlock resolution latency and defined in terms of 'n'- the number of transactions. From the table, it is worth noting certain points.

TABLE I COMPARISON OF VARIOUS VICTIM SELECTION POLICIES

Victim Selection Policy	Optimal in	Time Complexity
Youngest [5]	Fairness	O(n)
Min. History [5]	Fairness	O(n)
Least Static Priority [2]	Response time	O(n)
Maximum Size [8]	Throughput	O(n)
Min. no. of locks [5]	Resource Utilization	O(n)
Max. no of cycles [3]	Throughput	$O(n^2 \times k + 2)$ k– max. cycle size
Minimum abortion cost [11]	Resource Utilization	$O(n^3)$
Max. Edges [3]	Resource Utilization	$O(n^4)$
Blocker [5]	Resolution latency	O(1)
Initiator	Resolution latency	O(1)
Max. release set [4]	Resource utilization, throughput	$O(n^3m)$ m- no of resources
Min. work done so far [12]	Resource Utilization	$O(n^3)$
Priority + resource priority + min. work done [9]	Resource utilization, throughput	$O(n^4.m)$ m- no of resources

DOI: 10.5963/IJCSAI0204005

Youngest is fair in giving priority to the transactions based on their arrival time. So this will eliminate starvation in poverty [12]. But this might introduce starvation in wealth [13]. Static priority lets the user to configure the priorities of the transactions participating in the system. This eliminates starvation in wealth. This policy is ideal for real time systems. The transactions can be prioritized based on the need of immediate or delayed response time. But resolution using history eliminates both starvation in wealth and starvation in poverty. History based resolution is also fair in the sense that it does not penalize any transaction again and again.

Resolution based on transaction size will increase the number of transactions completed per unit time i.e. throughput.

Resolution policies like minimum number of locks, minimum work done and minimum abortion cost focus on lower rollback cost of a transaction, while algorithms like initiator, blocker, maximum edges and maximum release set choose a victim based on overall better performance of the system than on concerned individual transactions. The victims chosen using these algorithms might have already acquired all the required resources, completed maximum amount of execution or had been aborted again and again in the past. So they are not fair on individual transactions.

Blocker and initiator can be lower priority transactions and their only benefit is better deadlock resolution latency, especially in distributed systems.

While all the above mentioned algorithms abort one transaction per cycle, resolution based on maximum number of cycles tries to reduce this. So number of transactions executed per unit time i.e. throughput increases in this case.

A simulation experiment has been made to study their characteristics with respect to other attributes. To study these desirable characteristics, attributes of 500 transactions are randomly generated and tested. From the given victim selection algorithms, blocker and initiator algorithms are not considered because, they are optimal only in deadlock resolution latency and poor in other aspects. Victim selection algorithm by A.K. Srivastava and W. W. Shun [9] takes maximum deadlock resolution latency, hence it is also not considered.

In Fig. 1, it can be noticed that algorithm choosing victim based on maximum number of cycles provide maximum throughput i.e. more than 96%. This is because it aborts at most ‘n’ transactions, when there are ‘n’ cycles, whereas other algorithms abort atleast ‘n’ transactions. Then selection on maximum size provides better throughput i.e. 94%.This is because smaller transactions finish in time when the transactions are relatively smaller in size. In max edge algorithm, by aborting one transaction many transactions can proceed. Therefore the throughput is more in this case also. The performance of other resolution algorithms also depends on attributes of participating transactions and is bound to vary.

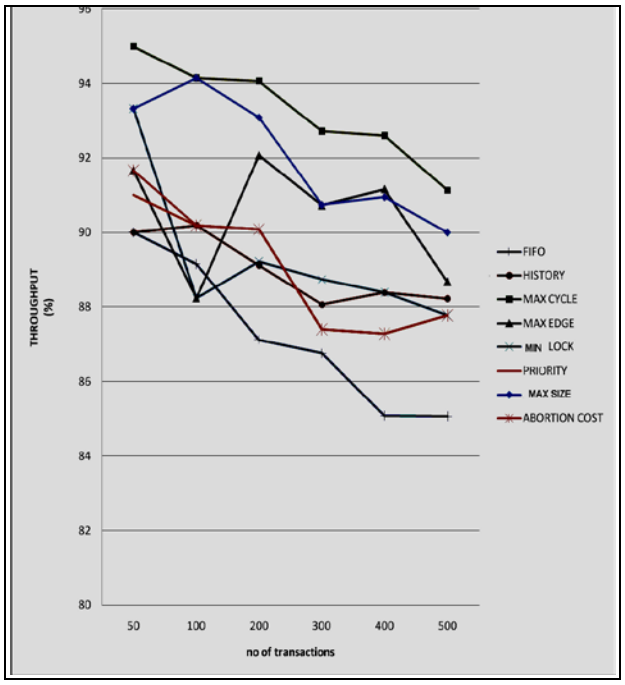

Fig. 1 Number of transactions versus throughput

In Fig. 2, resource utilization is compared by varying number of transactions. While throughput is measured in terms of number of transactions, resource utilization is measured in terms of resources. Maximum resource utilization happens when there is minimal rollback. Resource utilization is maximized in resolution algorithms of minimum number of locks and maximum size. It is also noticeable that minimum abortion cost based on arrival time and minimum number of operations has made the algorithm suboptimal in both aspects. But it is better than algorithms considering single transactional attribute.

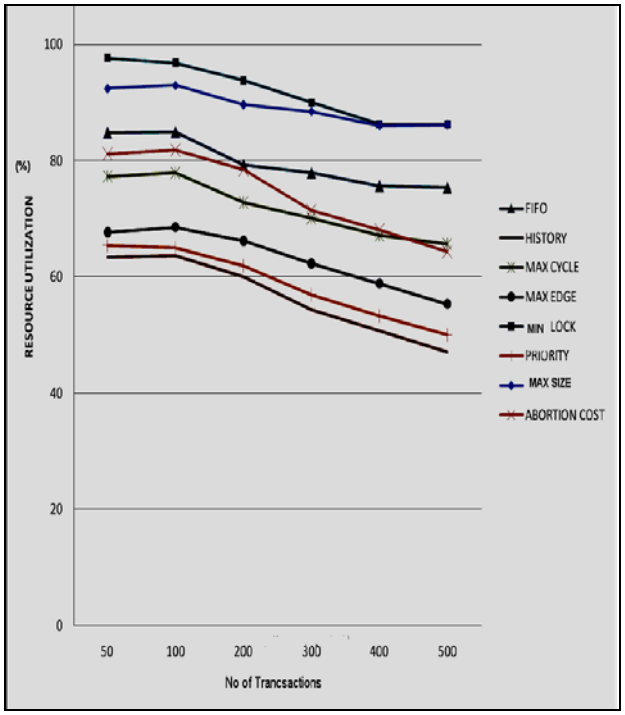

Fig. 2 Number of Transactions versus Resource Utilization

DOI: 10.5963/IJCSAI0204005

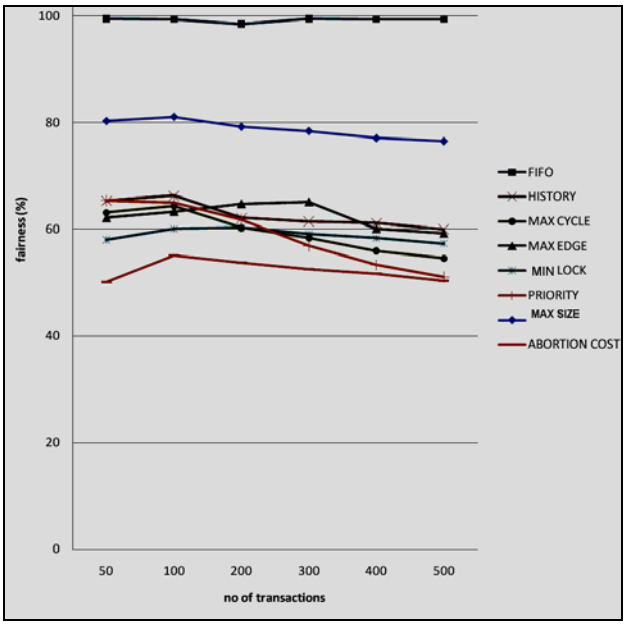

Fig. 3 Number of Transactions versus Fairness

In fig 3, fairness is compared with number of transactions. Fairness can be viewed in two aspects: based on age and starvation. The fairness considered in fig 3 is based on arrival time. While throughput, deadlock resolution latency and resource utilization are desirable attributes of the system, non- starvation and fairness are desirable attributes of individual transactions.

IV. PROPOSED ALGORITHM

The victim selection algorithms are based on transaction attributes and resource attributes. The common transaction attributes or characteristics are age, history, code size, priority, resource utilized so far and its attributes in WFG like in-degree, number of edges, out-degree and cycle participation. Apart from static priority, transactions are usually dynamically prioritized based on the attributes given above. Victim selection algorithm based on resource attributes could be based on resource priority, number of units of a particular resource, costly resource, resource most sought after etc.

The desirable attributes that could improve the system are higher throughput, better resource utilization and lesser deadlock latency. The desirable attributes in individual transaction execution is lower response time, fairness, no starvation and minimum roll back cost. It can be noticed that while each victim selection algorithm is optimal in one aspect, it is suboptimal in other aspects.

Hence the proposed algorithm is based on assigning weights which can be configured based on user requirement. For example in real time systems, response time is more important than other attributes. Similarly throughput is important in batch processing systems.

In the proposed algorithm, each transaction is expected to possess an attribute list to maintain its rank in various aspects. The attribute list of a transaction is as table IIA. The attribute list is created for every transaction arriving at the system. Attribute lists of all transactions whose execution are completed are deleted. Attribute lists of all live transactions whose execution are not completed, are also updated whenever a new transaction arrives or an old transaction leaves the system.

The rank of a transaction with respect to a particular attribute is based on its value relative to other transactions with respect to that factor. For example, if a transaction has arrived third among the active transactions in the system, then its rank with respect to age is fixed as 3. In general, the ranks are determined based on the seniority of the transaction.

TABLE II A TRANSACTION ATTRIBUTE LIST

Transaction ID	Rank
Age	
History	
No of resources requested	
No of resources granted	
Size	
Static priority	

TABLE II B DESIRABLE PERFORMANCE ATTRIBUTES OF THE SYSTEM

Desirable System Attribute	Wt as %	Rank of Transaction Attribute to Be Favored
Throughput	G	Size
Fairness	F	Age, history
Resolution latency	L	Initiator, random blocker
Response time	T	Static priority
Resource utilization	R	No of resources requested

The weights of desirable attributes in the system can be configured so that Σ (G, F, L, T, R) =100%(as in Table IIB). This is done based on the nature of the distributed system. The weights can be in the range 0 to 100%. The ranking of transactions in a centralized system is easy. However the deadlock detection and selection of a victim for resolution in distributed system is very tedious. The candidate victims are distributed in various sites. Selecting a victim transaction for abortion at each local site is suboptimal and affects the performance of the system. Hence global selection of victim needs propagation of transaction attributes to all sites. The sites in a distributed system communicate through messages. Hence probe based deadlock detection by K.M. Chandy and J.Misra [16] is one of the best distributed deadlock detection algorithms. In this algorithm, the transactions waiting for grant message will start sending probe message after time out. The probe is sent along the wait for edges of a Global Wait for Graph (GWFG). The probe message has the fields such as initiator (transaction initiating the probe), sender (transaction forwarding the probe), and receiver (transaction receiving the probe). This algorithm is used to detect cycle which indicates the presence of deadlock. Two new fields' namely current victim's transaction ID and its attribute list can be added to the probe for propagation for global victim selection. At each site, the rank of the current victim is compared with the locally selected victim. If the rank of local victim is higher than the current victim, the current victim can be replaced before forwarding the probe. When

DOI: 10.5963/IJCSAI0204005

the probe reaches the initiator, the Transaction ID which is to be aborted will be known. Then command can be sent to abort the victim to break the cycle and restart the system.

V. CONCLUSION

Resolution is an important phase in handling deadlocks. Selection of a victim influences the performance of the system strongly. The desirable performance parameters of the system like throughput, resource utilization are influenced by the victim selection. The desirable parameters of the transaction like fairness, resolution latency and response time are to be considered while selecting a victim. The weight based victim selection scheme balances the desirable parameters of transactions and system.

REFERENCES

[1] D.Zobel, C.Koch, Resolution Techniques and complexity results with deadlocks: A classifying and annotated bibliography, OS Review 22 (1), pp. 52-72, 1988.
[2] G.Newton, Deadlock prevention, detection and resolution: An annotated bibliography, ACM-SIGOPS OS Review, vol. 13, no. 4, pp. 33-44, 1979.
[3] Mukesh Singhal, Deadlock Detection in Distributed Systems, IEEE survey and tutorial series, Vol. 22, pp.37-48, 1989.
[4] A.Moon, H.Cho, Performance Analysis of Global Concurrency Control Algorithms and Deadlock Resolution Strategies in Multi database Systems, IEEE, 1997.
[5] R.Agarwal, M.Carey and L.Mcvoy, The performance of Alternative Strategies for dealing with Deadlocks in database Management Systems, IEEE Transactions on Software Engineering, SE-13 (12), pp. 1348-1363, 1987.
[6] M.Sinha and M.Natarajan, A Priority based Distributed Deadlock Detection Algorithm, IEEE transactions on Software Engineering, vol. SE-11, pp.67-80, 1985.
[7] G. Weikum and G.Vossen, Transactional Information Systems, ACM, 2005.
[8] Y.Chow, W.F. Klostermeyer and K.Luo, Efficient techniques for Deadlock Resolution in Distributed Systems, IEEE, 1991.
[9] M.Garey and D.Johnson, Computers and Intractability: A Guide to the Theory of NP Completeness, W.H. Freeman and Company, New York, 1979.
[10] I.Terekov and T.Camp, Time efficient deadlock resolution algorithms, Information Processing Letters, Elsevier Science, pp.149 -154, 1999.
[11] Alok Kumar Srivastava and Wilson Wai Shun, Victim selection for deadlock detection, United States Patent, US7185339B2, 2007.
[12] Richard Holt, Some deadlock properties of Computer Systems, ACM Computing Surveys, Vol. 4, No. 3, pp. 179-196, 1972.
[13] D.L. Parnas and A.N. Habermann, Comment on deadlock prevention method, Communications ACM, vol. 15, no. 9, pp. 840-841, 1972.
[14] Xuemin Lin and Jian Chen, An Optimal Deadlock Resolution Algorithm in Multidatabase Systems, Proceedings of ICPADS, 1996.
[15] X.Lin, M.E. Orlowska & Y.Zhang, An Optimal Victim Selection Algorithm for removing Global Deadlocks in Multidatabase Systems, 9th International Conference on Frontiers of Computer Technology, FCT'94, IEEE CS Press, pp. 501-505, 1994.
[16] K.Mani Chandy and Jayadev Mishra, Distributed Deadlock Detection, ACM Transactions on Computer Systems, Vol.1, No.2, pp. 144-156, 1983.

DOI: 10.5963/IJCSAI0204005