Resolution Based on MRAA for Generalized Distributed Deadlocks

V.Geetha
Department of Information Technology
Pondicherry Engineering College
Puducherry, India.
vgeetha@pec.edu

S.Batmavady
Department of Electronics &Communication Engineering
Pondicherry Engineering College
Puducherry, India.

Abstract—A deadlock occurs when there is a cycle of processes waiting for the resources held by others. This deadlock can be resolved only when the cycle is broken i.e. when one of the processes is victimized. None of the previous resolution algorithms choose a victim based on the work done by it before it is aborted. When it aborts, it cancels its pending requests and releases all the resources it had. The aborted process has to restart all the requests to complete its work. The performance of the system deteriorates as a process that has most of its resources allocated is often chosen as a victim. Hence in the proposed algorithm, a victim is chosen based on the Minimum Resource Allocated Already (MRAA) principle. This minimizes the wasted work done and hence the performance of the distributed systems is improved.

Keywords- ***Distributed systems, Resolution, Generalized model, resource deadlocks***

I. Introduction

In distributed systems, deadlock detection and resolution is a major problem. All distributed deadlock detection and resolution algorithms are based on edge chasing algorithm [1] and use Wait For Graph (WFG), Asynchronous Wait For Graph (AWFG), Colored Wait For Graph (CWFG), Resource Allocation Graph (RAG) to model the dependencies. The description of these models is given in Jose et al [6]. The edge chasing algorithm proposed by chandy et al [1] uses probes to detect deadlocks. A deadlock is detected when the message comes back to its initiator.

All Distributed Deadlock Detection algorithms must satisfy the correctness criteria namely

- Liveliness: Every dead lock is resolved in finite time
- Safety: the algorithm does not resolve false deadlocks.

Further there should be no spontaneous abortions. The proposed algorithm can be applied for resource deadlocks in which the process can execute only after getting all resources. This algorithm supports generalized model which is a combination of OR and AND requests. It supports SR (Single Request) model which means the process can request for only one resource at a time.

In this paper we focus on improving the performance of the distributed system by choosing a victim, which has done minimum work, i.e. which has got minimum resources allocated to it. This resolution algorithm can be used with any of the detection algorithms with a slight modification in the probe model.

A. Previous Works

Several distributed deadlock resolution algorithms have been proposed in the previous works. In all these algorithms the victims are chosen to minimize the number of messages i.e. they aim at minimizing the communication cost. All these algorithms aim at removing phantom deadlocks and reduce the deadlock latency. These algorithms aim at preventing simultaneous probe initiations for the same deadlock and hence try to avoid phantom deadlock deductions. So in these algorithms there will be only one lively probe for every deadlock occurrence. They try to do this by using timestamps or prioritized instances of the same algorithm.

In all these existing works the victim is either killed [2, 4, 6, 7,8, 9,10,12] or forced to suicide [3,5,11,12,13].The victim is either initiator [3,4,5] or lowest priority node [2,6,7,8,9,10,13] or deadlock detecting node[12]. The priority is pre-fixed or dynamically assigned. There is a chance of the initiator or low priority node having most of its resource requests granted. None of them consider the performance overhead involved in victimizing a node which has got most of its resource allocated. When a node which has most of its requests granted, is aborted, all the work done by the node so far is cancelled. This node has to restart the requests again later to get all its resources.

Hence the proposed algorithm aims at providing better performance by combining with low communication cost provided by the existing deadlock detection algorithms.

B. Contributions of the Paper

In the proposed algorithm, the deadlock resolution is done by choosing a victim based on Minimum Resources Allocated Already (MRAA) principle. This can be done by doing slight modifications in the processes' data structures and Probe model.

The process is modified to calculate the work done (WD) which is the ratio of count of the resources allocated to it so far to resources required.

When deadlock occurs and a probe is initiated, the probe model is altered to have two more tags namely the victim Process id and WD of the node which has the MRAA so far. Initially it has the id of the initiator or the node which detected the deadlock and it's WD. As the probe is forwarded in the dependent set, this will be updated to have the id of the node which has MRAA and its work done in the dependent set it has traveled so far. When more than one process has the WD lowest and equal, the first node that has the lowest WD is aborted. This reduces the communication cost.

When the cycle is detected, i.e. when the probe message returns back to the initiator, it will have the id of the process which has minimum WD among the nodes in the dependent set involved in the deadlock. Then a message is sent to the victim to abort itself. The victim aborts itself by rolling back all the resources allocated.

C. Overall Organization

The organization of the paper is as follows. Section 2 describes the system model. Section 3 describes the features of the proposed algorithm. Finally conclusions and references end the paper.

II. System model

The following assumptions are made in the proposed system.

The distributed system can be viewed as a collection of sites at which transactions and resource managers are executed. The resource managers maintain the reusable resources. The unblocking function of generalized transactions will be of the form $f_T = (p \wedge q) \vee r$, where p, q and r are resources. The function f_T indicates that the transaction will be in executing state only if it is true. The REQUEST, GRANT, RELEASE and CANCEL are computational messages generated during the execution of the transactions. The PROBE and ABORT are control messages sent during deadlock conditions to detect and resolve them. The nodes may be idle (blocked) or executing (active or unblocked). When they are in executing state, they can send both computation and control messages. When they are idle they can send only control messages. In a SR model, an active transaction makes a request to a resource manager and enters into blocked state. It becomes active again only after getting the resource granted. This procedure is continued until it gets the 'm' resources out of 'n' requests in generalized model. If it does not get sufficient replies from the resource managers within a stipulated time, it suspects deadlock and issues probe to verify it.

The distributed system is modeled using logical structures like WFG. The incoming edges to a node have the identifiers of the processes that have dependency edges incident on it. These are from the resource managers which has granted resources to this node. Thus in degree of every node indicates the resources allocated to the process so far. This can also be computed by counting the number of grant messages received so far. The out degree of any node will be usually one for SR model. This will point to the dependency of this node on another node.

All processes have a system wide unique id. There is no shared memory in the system. The processes communicate by message passing. We assume that communication channels are reliable. i.e. messages are neither lost nor replicated and support error free transmission. The messages are received in the order in which they are sent. The message delays are arbitrary but finite.

A. Data structures and messages

The following is the description of the types of messages that processes may send to each other:

- REQUEST: sent by a transaction to a resource manager to request the resource managed by it.
- GRANT: sent by resource manager to the transaction to grant the access of resource managed by it.
- RELEASE: Sent by the transaction to release the resource to the resource manager.

- CANCEL : to withdraw the request it has made (used in OR requests)
- PROBE: message initiated by a transaction that suspects a deadlock and is forwarded by various processes to detect deadlock.
- ABORT: Sent by the initiator to abort the victim.

B. Description of the Algorithm

The transaction process T may require ‘m’ out of ‘n’ resources as it supports generalized model. Initially it is in ‘executing’ state. It has to send ‘n’ request messages to various resource managers which are controlling those resources. But it is enough to get ‘m’ replies, m being a subset of ‘n’, to execute the transaction. There is going to be counter called RGcount (Resource Granted count) in every node, which will be incremented every time it receives a GRANT message. Initially it will be zero and the maximum value can be ‘m’.

```
Data structure of Probe
Begin
Initiator_ ID: Integer;
Forwarding_ process_ID : Integer;
Receiving _Process_ID : Integer;
Victim_ID: Integer;
VictimWD: Floating point;
/* other fields used in the chosen detection algorithm
may be added here */
End
Data structure of node
Begin
ID: Integer; RGcount : Integer;
M: Integer;
N: integer;
Workdone: Floating point;
/* other fields used in the chosen detection algorithm
may be added here */
End
```

Figure 1. Data structure of probe and node.

The probe is usually of the format P (i, j, k), where i is the id of the initiator, j is the id of the forwarding process and k is the id of the receiving process. Apart from this, additional fields will be present in all the detection algorithms to minimize the deadlock detection and resolution latency. Along with them, now two more fields are attached to the probe namely i) victimid:. id of the process which has done lowest work done so far and ii)wdv: its work done ratio. So now the format of the Probe will be P (i, j, k, vid, , wdv,….).When a probe is initiated either by itself or forwarded by other initiators reaches it, the node will compute the work done by it.

Work done is the ratio of *resources allocated already (RGCount)* to the *number of resources required (m) to complete the transaction.*

Hence work done (wd) = RGcount / m

Figure 1. shows the data structures of probes and nodes. Since the formats of other messages are not altered, they are not given

Initial values of these fields will be the initiator id and its work done. As the probe traverses through the WFG, this will be replaced by the id and work done of the node which has done minimum work so far in the path. This is done by comparing the work done of the victim so far in the probe format with itself. If the work done by the current node is less than the work done by the victim so far, it will replace its id and work done in victimid and victimwd fields and forwards it. Else it simply forwards the probe. Hence when a deadlock is detected, the probe that returns to the initiator will have the id of the node which has lowest work done in the cycle. This is shown in figure 2.

the probe format with itself. If the work done by the current node is less than the work done by the victim so far, it will replace its id and work done in victimid and victimwd fields and forwards it. Else it simply forwards the probe. Hence when a deadlock is detected, the probe that returns to the initiator will have the id of the node which has lowest work done in the cycle. This is shown in figure 2.

```
Initial conditions:
For all i Є nodes in the distributed system
state($node_i$) ← active;
RGcount = 0; Workdone = 0.0
When node i receives GRANT message
Begin
State ← active; RGcount = RGcount + 1;
/* additional code to reduce latency*/
End
When node i initiates probe message
Begin
VictimID = ID; Workdone = RGcount/M;
VictimWD = Workdone;
/* additional code to reduce latency*/
Initiate (probe);
End
When node i initiates probe message
 Begin
Workdone = RGcount / M;
If Workdone < VictimWD then VictimWD = workdone;
VictimID = ID;
/* additional code to reduce latency*/ Forward (Probe);
End;
When node i receives probe message
(after detecting deadlock)
Begin
Abort(victimID);
End;
```

Figure 1. Pseudo code of the proposed algorithm.

III. CONCLUSION

In a distributed system it is not enough to reduce the deadlock detection and resolution latency. The objective should be to maximize the performance while minimizing the communication cost. The proposed algorithm aims at maximizing the performance of the distributed systems, while trying to reduce the deadlock latency also. The deadlock resolution is done in such a way that the minimum work done node is penalized so that previous work done is not wasted. Several deadlock detection algorithms exist for generalized model and [13] provides a O (n) solution for it. When this proposed algorithm is combined with those algorithms the distributed system will give maximum throughput.

References

[1] K.Mani Chandy and Jayadev Misra, “Distributed Deadlock Detection”, ACM transactions on computer systems, vol. 1, No. 2, pages144-156,1983.

[2] Marina Roesler and Walter.A.Burkhard, “ Resolution of Deadlocks in Object-oriented Distributed Systems”, IEEE Transactions on Computers, Vol. 38, No. 8, August 1989.

[3] Shigang Chen, Yi Deng, Wei Sun and Naphtali Rishe, ”Efficient Algorithms for Detection and Resolution of Distributed Deadlocks” IEEE,1995.

[4] Young Man Kim, Ten Hwang Lai and Neelam Soundarajan, ”Efficient Distributed Deadlock Detection and Resolution Using Probes, Tokens and Barriers”,IEEE,1997.

[5] Jesus Villadangos, Federico Farina and Jose R.Gonzalez de Mendivil, “A Safe Distributed Deadlock Resolution Algorithm for the OR Request model”, IEEE, 1998.

[6] Jose Ramon Gonzalez de Mendivil, Jose Ramon Garitagoitia and J.M. Bernabeu- Auban, “A Distributed Deadlock Resolution Algorithm for the AND Model”, IEEE, 1999.

[7] Soojung Lee,” Efficient Generalized Deadlock Detection and Resolution in Distributed Systems”, IEEE, 2001.

[8] Soojung Lee,” Fast Detection and Resolution of Generalized Distributed Deadlocks”, Proceedings of 10th Euromicro workshop on parallel, Distributed and Network based Processing, IEEE, 2002.

[9] Jesus Villadangos, Federico Farina, Jose Ramon Gonzalez de Mendivil and Jose Ramon Garitagoitia, “A Safe Algorithm for Resolving OR Deadlocks “, IEEE, 2003.
[10] Soojung Lee, “Fast, Centralized Detection and Resolution of distributed Deadlocks in the Generalized Model”, IEEE, 2004.
[11] Mehdi Hashemzadeh, Nacer Farajzadeh and Abolfazl T. Haghighat,” Optimal Detection and Resolution of Distributed Deadlocks in the Generalized Model”. IEEE, 2006.
[12] Nacer Farajzadeh , Mehdi Hashemzadeh, Morteza Mousakhani and Abolfazl T. Haghighat, “An Efficient Generalized Deadlock Detection and Resolution Algorithm in distributed Systems”,CIT’05,IEEE, 2005.
[13] Manuel Prieto, Jesus Villadangos, Federico Farina and Alberto Cordoba,”An O(n) distributed deadlock resolution algorithm”, PDP’06, IEEE, 2006.