International Journal of Advances in Engineering and Management (IJAEM)
Volume 2, Issue 10, pp: 12-15 www.ijaem.net ISSN: 2395-5252

An Extensive Analysis of Deadlock Resolution Algorithms

Kamala PL, Sri Abinaya S, Swarnalakshmi D, Dr. M. Sujithra M.C.A., M.Phil., Phd, Dr.A.D. Chithra M.C.A., M.Phil., Phd
2nd Year M.Sc Software Systems (Integrated) Coimbatore Institute of Technology, Coimbatore.
Assistant Proferssor, Department of Data Science Coimbatore Institute of Technology,Coimbatore.
Assistant Proferssor,Department of Software Systems Coimbatore Institute of Technology, Coimbatore.

Date of Submission: 20-11-2020 Date of Acceptance: 03-12-2020

ABSTRACT: A deadlock is a situation where set of processes waits for resources held by other processes in the same set. This phenomenon implies disaster in man-made systems; therefore, it must be carefully handled by system designers, analysts and engineers. The processes in deadlock waits indefinitely for the resources and the resources held by these processes are not available for any other process. These processes never terminate their execution. Sometimes it may lead to a serious system failure. The deadlock is resolved using a Deadlock resolution algorithm. The primary step is to select the victim, then evict/abort the victim or cause it to rollback. This step resolves the deadlock easily. This paper describes wait for graph (deadlock detection) and some deadlock resolutionalgorithms to resolve the deadlock using different criteria.

KEYWORDS: Deadlock, Resources, Processes, Release, Deadlock prevention, Deadlock resolution, wait for graph, Deadlock handling, VGS Algorithm.

I. INTRODUCTION:

Deadlock occurs when no process proceeds for execution, each waits for resources that has been taken by other processes. For example, one can't the job without having the (professional) experience and one can't get the experience without having a job.

Traffic gridlock is an everyday example of a deadlock situation. There is a variant of deadlock called livelock. It is important to note that the number of processes and the number and kind of resources possessed and requested are unimportant. The resources may be either physical or logical. Examples of physical resources are printers, tape drivers, memory space and CPU cycles. Examples of logical resources are files, semaphores and monitors.The sequence in which the process utilizes a resource is Request, Use and Release. Deadlock can be detected by the wait-for-graph. A deadlock needs to be resolved timely otherwise, the deadlock size will increase. The deadlock size is defined as the total number of blocked processes (BP) involved in deadlock, where BP refers to processes that waits indefinitely on resources acquired by other processes. Deadlock can be resolved by avoiding at least one of the four conditions, mutual exclusion, hold and wait, no pre-emption and circular wait. Because, all these four conditions are required simultaneously to cause deadlock.

PROBLEM SPECIFICATION:

Most of the deadlock resolution algorithms imply rollback/abort as the solutions to deadlock. The only difference lies in the criteria for selecting victim. In this case, evicting the victim, or restarting the victim leads to wastage of resources, wastage of work done by the aborted processes, low throughput of system and the process execution time becomes unpredictable. Restarting a process is more expensive than waiting, thus aborting the victim needs to be avoided.

Therefore, in this paper an algorithm that do not cause any aborts/rollbacks is proposed. It resolves the deadlock with the mutual cooperation of the transactions.

DEADLOCK:

A deadlock is a situation in which every process of a group, is waiting for another process, including itself, to acquire action, i.e., releasing a lock.

International Journal of Advances in Engineering and Management (IJAEM)
Volume 2, Issue 10, pp: 12-15 www.ijaem.net ISSN: 2395-5252

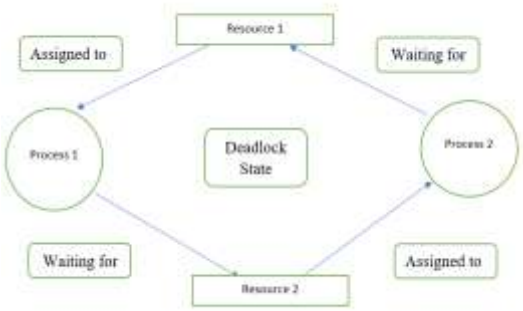

Figure 1. Deadlock in Operating Systems

It is a common problem in multiprocessing systems, parallel computing. In a communication system, deadlocks occur mainly due to lost or corrupt signals rather than resource conflict.
If the resources aren't available at that point , the method enters a wait state. Waiting processes may never turn again because the resources they need requested are held by other waiting processes. This situation is called deadlock. A process requests a resource before using it, and releases resource after using it.
1. Request: If the request is not permitted immediately then the requesting process waits until it acquires the resource.
2. Use: the method can operate the resource
3. Release: The process releases the resource.

METHODS FOR HANDLING DEADLOCK:

1. Process Termination:

To eliminate the deadlock, we will simply kill one or more processes. For this, we use two methods:
(a) Abort all the Deadlocked Processes:
Aborting all the processes will definitely break the deadlock, but with an excellent expense. The deadlocked processes may have computed for an extended time and therefore the results of those partial computations must be discarded and there's a probability to recalculate them later.

(b) Abort one process at a time till eliminating deadlock:
Until deadlock cycle is eliminated from the system, abort one deadlocked process at a time,. This method, there can cause considerable overhead, because after aborting each process, there is a run deadlock detection algorithm to check if any processes is still deadlocked.
2. Resource Pre-emption:
(a) Selecting a victim:
The resources and processes that are to be pre-empted and also the order to minimize the cost, are determined
(b) Rollback:
We must determine what should be done with the process from which resources are pre-empted. One simple idea is total rollback. That means abort the process and restart it.

(c)Starvation:
In a system, it should happen that very same process is usually picked as a victim. Finally, that process will never complete its assigned task. It is called Starvation and must be prevented. One solution is that a process must be picked as a victim only a fixed number of times.

II. RESOLUTION ALGORITHM:

A. RESOLUTION BY USING TIMESTAMP

The criteria chosen for aborting the victim is based on timestamp. Every process are assigned with a timestamp when they enter the system. The timestamp of the younger process is greater than that of the older process. The process with a higher timestamp i.e., younger process is aborted first. The reason for choosing process is that it would have consumed less resources and less CPU time.
The problem here is that, the it causes starvation, because every time aborting a younger process can lead to its starvation, preventing its completion.

B. RESOLUTION BY USING BURST TIME

The criteria chosen for aborting the victim is based on the burst time of the processes. Burst time is the CPU time needed by a process for its execution. To break the deadlock cycle, the process with maximum burst i.e., older process is aborted first. The problem here is that, the older process would have consumed more resources and been in the system for very long, but still it doesn't complete its execution. This is an inefficient approach.

C. RESOLUTION BY DEGREE

The criteria chosen for aborting the victim is based on the degree of the processes. In wait – for – graph approach, the degree determines how many resources a process is holding and how many process a resource is requesting. There are two types of degrees. In-degree: denotes the number of resources held by a process. Out-Degree: denotes the number of

International Journal of Advances in Engineering and Management (IJAEM)
Volume 2, Issue 10, pp: 12-15 www.ijaem.net ISSN: 2395-5252

resources requested by a process. Degree of a process is calculated by taking sum of in-degree and out-degree. The process having highest degree is aborted first.

D. RESOLUTION BY BOTH TIMESTAMP AND BURST TIME

The younger process with higher burst time is chosen as the victim.

The advantage of choosing this approach is that younger process which takes maximum burst time is aborted and it allows processes with less execution time to complete first.

E. RESOLUTION BY BOTH BURST TIME AND DEGREE

The process with highest burst time and highest degree is chosen as the victim i.e., the process having more resource request and high time to complete will be aborted. The advantage of choosing this approach is that choosing processes 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 BOTH DEGREE AND TIMESTAMP

A younger process with higher degree will be aborted. The problem of starvation which occurs when selecting a process based on timestamp will be avoided in this case as the degree of the node is also considered along with timestamp so as to pick victim for resolving deadlock within the system.

G. VGS ALGORITHM FOR DEADLOCK RESOLUTION

It is an efficient deadlock resolutionalgorithm. This algorithm is based on themutual cooperation of transactions and isdescribed as follows:

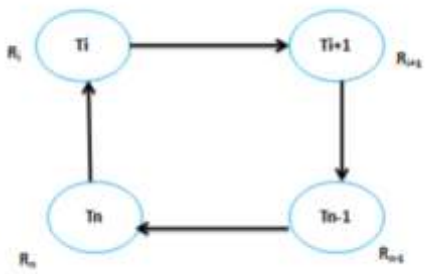

Figure 2. A Deadlock cycle

Suppose Ti, Ti+1, Ti+2……Tnare the transactions involved in a deadlock.They form a deadlock cycle such that Tiholds resource Ri, Ti+1 holds resourceRi+1, Ti+2 holds resourceRi+2……..Tn holds Rn and Ti isrequesting for resource Ri+1 , Ti+1 isrequesting for resource Ri+2 ……,Tn isrequesting for Ri. Since each transaction isholding a resource and waiting indefinitelyfor other resource held by the othertransaction, they form a deadlock cycle andnone of them is being able to proceedahead.In the proposed deadlock resolutionalgorithm transaction, coordinator observesthe scenario and it suspends Ti+1 for somerandom t seconds and it releases resourceRi+1 which is acquired by the requestingPi Pj Pk transaction Ti. For t seconds it has been alloted a resource which is the time periodin which Ti+1 has been suspended. Ti issupposed to utilize Ri+1 and execute successfully in t seconds. If Ti successfullyexecutes before t seconds it sends amessage to coordinator that it hassuccessfully executed and to resumetransaction Ti+1 and gives its resourceRi+1 back to Ti+1. If Ti is not able tocomplete its execution within t secondcoordinator pre-empts resource Ri+1 fromTi and provides it back to Ti+1. The valueRi+1 is the value partially updated by Ti.Now Ti+1 will check whether Ti is stillrequesting for Ri+1. If it is requesting, Ti+1informs coordinator and is suspended againfor some random t seconds and resourceTi is alloted to Ri+1 again andTi receives itand resumes its execution and whencompleted before t seconds Ti informscoordinator to resume Ti+1 and gives backresource Ri+1 to Ti+1.

III. ALGORITHM:

TRANSACTION (Ti, Ti+1….Tn),
RESOURCE (Ri, Ri+1……Rn)
START:
Suppose Ti……...Tn be the transactionsinvolved in a deadlock and form a cycle.
BEGIN,
Ti holds resource Ri, Ti+1 hold resourceRi+1…Tn holds resource Rn andTi requests resource Ri+1,Ti+1 requestsresource Ri+2…Tn requests resource RiEach transaction is in a circular wait andhold condition
DO,Coordinator suspends transaction Ti+1 andTn for random t seconds and releasesresource Ri+1 and Rn respectively.
{
Ri+1 is now taken by transaction Ti and itexecutes.
{
IF Ti executes successfully before t seconds
{{

Ti informs coordinator to resume Ti+1, Ti+1 resumes and takes the resource Ri+1back.
}
Now Ti+1 will wait for resource Ri+2 andwill proceed successfully as there is nodeadlock now
}ELSE{
Ti+1 preempts the resource from Ti and
value of Ri+1 will be the value partially
updated by Ti
}
Ti+1 CHECKS
IF
Ti is still requesting for resource Ri+1{{
Coordinator again suspends Ti+1 forrandom t seconds andgives the resource Ri+1 to Ti, it will acquire the resource Ri+1 and willlock it. After Ti executes successfully itreleases Ri+1. Ti informs coordinator toresume Ti+1 and gives its resource Ri+1back
}
Now Ti+1 will wait for resource Ri+2 andwill proceed successfully as there is nodeadlock now
}
ELSE Ti+1 will wait for resource Ri+2 andwill proceed successfully as there is nodeadlock now
}

IV. CONCLUSION:

In this paper we presented differentdeadlock resolution algorithms whichresolves deadlock. A detailed explanationof VGS algorithm which solves deadlockeffectively is explained. As the paperdescribes in this algorithm the transactionsresolve deadlock with the mutualcooperation of each other. Transaction Ti+1and Tn suspend themselves and let othertransactions proceed successfully andcontinuously co-operate them till they arenot able to commit successfully. Thisalgorithm does not cause anyabortion/rollback, which shows itseffectiveness. In the proposed algorithm thecoordinator manages its own transactionsand resolves any deadlock if detected.

REFERENCE:

[1]. M. Singhal, "Deadlock Detection in Distributed Systems, "Computer, vol, 40, no. 8, pp. 37-48, Nov. 1989.
[2]. SeemaPayal, Ruchi Taneja, "Deadlock – A problem of Computer system, "Volume 5, pp.413-416, Aug 2015.
[3]. Sumika Jain, Nitin Kumar, Kuldeep Chauhan, "An overview on Deadlock Resolution Techniques, " Volume 7, Issue 12, pp.1-3, 2019.
[4]. PoojaChahar, Surjeet Dalal, "Deadlock Resolution Techniques: An Overview, vol 3, Issue 7, July 2013.
[5]. Kunwar Singh, Menka Goswami, Ajit Singh, " VGS Algorithm – an Efficient Deadlock Resolution Method, vol 44-No.1, April 2012.