International Journal of Science and Research (IJSR)
ISSN: 2319-7064
SJIF (2019): 7.583

The Deadlock Problem - A Review

Naveen Dubey

Scholar, CSE Department PDM University, Bahadurgarh, Haryana, India

Abstract: ***Deadlock is a phenomenon in which a system or a part of it remains indefinitely blocked and cannot terminate its task. Such phenomenon often implies disaster in man-made system and, therefore must be carefully handled by system designers, analysts and engineers. Computer systems are prone to deadlock. Deadlock is a result to some uncontrolled sequence of release and request of resource among processes in a system. This survey paper presents some system models and deadlock handling techniques to deal with the problem. Selected algorithms are also presented to see how deadlocks can be deleted. In this paper, we are going to presents several algorithms that handle deadlock in a system. A deadlock can be resolved by aborting one or more processes in the deadlocked-set and is released, and withdraw all the resource requests it has made.***

Keywords: Deadlock, Condition for deadlock, Deadlock prevention, Deadlock avoidance, Deadlock detection, Deadlock handling, Deadlock recovery, Banker's Algorithm

1. Introduction

A set of process is in a deadlock state if each process in the set is waiting for an event that can be caused by only another process in the set. In other words, each member of set of deadlock processes is waiting for a resource that can be released only by a deadlock process. None of the processes can run, none of them can release any resources, and none of them can be awakened. It is important to note that the number of processes and the number and kind of processed and requested are unimportant.

The resources may be either physical or logical. Examples of physical resources and printers. Tape Drivers, Memory Space and CPU Cycles. Examples of logic resources and files, Semaphores, and Monitors.

A process May utilize a resource in the sequence Request, Use and release. These operations are accomplished by wait and signal. A set of process is in deadlock state when every process in the set of waiting for an event that can be caused only by another process in the set. Deadlock detected by the wait -for-graph. Deadlock is removed by different mechanisms like deadlock prevention, deadlock handling and deadlock recovery.

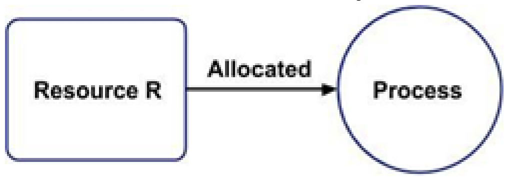
2. Explanation

Deadlock Conditions

Process do not run constantly from the time they are created until their termination; they can be identified in three different state; ready, running and blocked. At the ready state, a process is stopped, allowing some other process to run, at the running state, a process is utilizing some resource, and at the blocked state, the process id stopped and will not start running again something trigger it to restart.

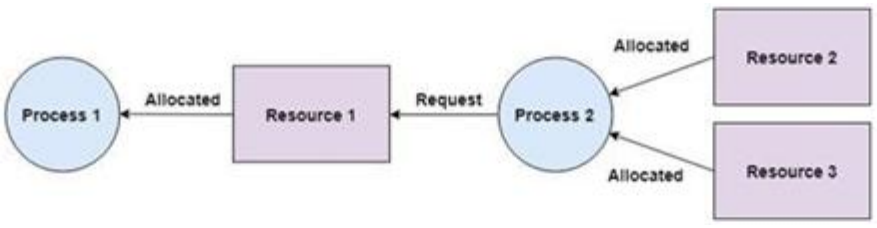
1) Mutual Exclusion Condition

There should be a resource that can be held by one process at a time. In a diagram below, there is a signal instance of Resource 1 and it is held by Process 1 only.



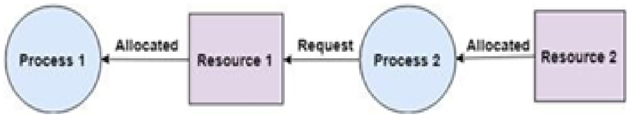
2) Hold and Wait

A process can hold multiple resources and still request more resources from other processes which are holding them. In the Process 2 holds resource 2 and Resource 3 and is requesting the resource 1 which is held by process 1.



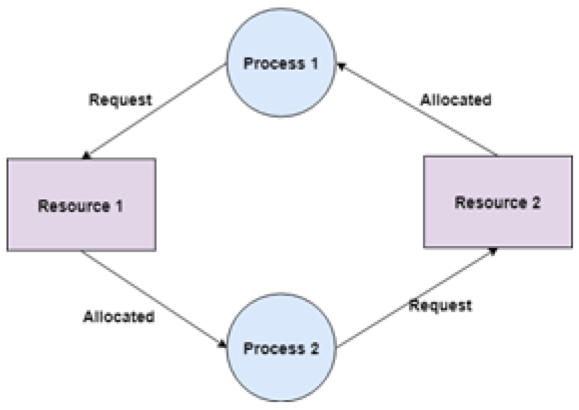
3) No Preemption

Resources cannot be preempted from a process by force. A process can only released a resource voluntarily. In the diagram below, Process 2 cannot preempt Resource 1 from Process 1.It will only be released when process 1 relinquishes it voluntarily after its execution is complete.



4) Circular Wait

A process is waiting for the resource held by the second process, which is waiting for the resource held by the third process and so on, till the last process is waiting for a resource held by the first process. This forms a circular chain. For example: Process 1 is allocated Resource 2 and it is requesting Resource 1. Similarly, Process 2 is allocated Resource 1 and it is requesting Resource 2. This forms a circular wait loop.



<u>How to deal with Deadlock?</u>

In general, there are four strategies of dealing with deadlock problem.

1) The Ostrich Approach
2) Just ignore the deadlock problem altogether.
3) Deadlock Prevention
4) Prevent deadlock by resource scheduling so us to negate at least one of the four conditions.
5) Deadlock Avoidance
6) Avoid deadlock by resource scheduling.
7) Deadlock Detection and Recovery
8) Detect deadlock and when it occurs, take steps to recover.

The Ostrich Approach

In this approach, the deadlock is ignored and user continues its working.

Deadlock Prevention

By ensuring that at least one of these conditions cannot hold, we can prevent the occurrence of a deadlock.

Elimination of "Mutual Exclusion" Condition

The mutual exclusion must hold only for non-shareable resource. That is, several processes cannot simultaneously share a single resource. This condition is difficult to eliminate because some resource such as the tap drive and printer, are inherently non-shareable Note that shareable resources like read-only-file do not require mutually exclusive access and thus cannot be involved in deadlock.

Elimination of "Hold and wait" Condition

There are two possibilities for elimination of the second condition. The first alternative is that a process request be granted all of the resources it needs at once, prior to execution. The second alternative is to disallow a process from requesting resources whenever it has previously allocated resource. This strategy requires that all of the resource a process will need must be requested at once. The system must grant resource on "all or none" basis. If the complete set of resource needed by a process is not currently available, then the process waits until the complete set is available. While the process waits, however, it may not hold any resource. Thus the "wait for" conditions are denied and deadlocks simply cannot occurs. This strategy can lead serious waste of resource.

Elimination of "No-Preemption" Condition

The non-preemption condition can be alleviated by forcing waiting for a resource that cannot immediately be allocated to relinquish all of its currently held resource so that processes may use them to finish. Suppose a system does allow process to hold resource while requesting additional resource.

Consider what happens when a request cannot be satisfied. A process holds resources a second process may need in order to proceed while second process may hold the resource needed by the first resource. This is a deadlock. This strategy requires that when a process that is holding some resource is denied a request for additional resources. The process must release its held resources and, if necessary, a request then again together resource, Implementation of this strategy denies the "no-preemption" condition effectively.

High cost when a process release resource the process may lose all its work to that point. One serious consequence of this strategy is that possibility of indefinite postponement (starvation).A process might be held off indefinitely as it repeatedly requests and release the same.

Elimination of "Circular wait" Condition

The last condition, the circular wait, can be denied by imposing a total ordering on all of the resource types and than forcing, all processes to request the resource in order(increasing or decreasing).This strategy impose a total ordering of all resource types, and to require that each process requests resources in a numerical order of enumeration. With this rule, resource allocation graph can never have a cycle.

Deadlock Avoidance

In deadlock avoidance, the request for any resource will be granted if the resulting state of the resource doesn't cause deadlock in the system. The state of the system will continuously be checked for the unsafe states.

In order to avoid of avoid deadlock, the process must tell OS, the maximum number of resource a process can request to complete its execution.

The simplest and most useful approach states that the process should declare the maximum number of resource of each type it may ever need. The deadlock avoidance algorithm examines the resource allocations so that there can never be a circular wait condition.

Safe State: A state is safe if the system can allocate resource to each process in some order and still avoid a deadlock. A system is in safe state only if there exists a safe sequence. A safe sequence of process is a safe sequence if it can allocate the resource to theprocesses.

Unsafe State: A deadlock state is unsafe state. An unsafe state may lead to a deadlock. In an unsafe state, the operating system cannot prevent processes from requesting resources.

Banker's Algorithm
This is a deadlock avoidance algorithm. The was chosen because this algorithm could be used in a banking system to ensure that the bank never allocates. Its available cash such that it can longer satisfy the needs of all customers. When a new process enters the system, it must declare the maximum of instances of each resource type that it may need. This number may not exceed the total number of resources will leave the system in a safe state. If it will the resource are allocated; otherwise, the process must wait until some other release enough resources.

In this analogy Customers=process Units=resource, say, tape, drive Banker=Operating System

Customers	Used	Max
A	0	6
B	0	5
C	0	4
D	0	7

Available Units=10
Figure 1

We see four customers each of whom has been granted a number of credit units. The banker reserved only 10 units than 22 units to service them. At certain moment, the situation becomes.

Customers	Used	Max
A	1	6
B	1	5
C	2	4
D	4	7

Available Units=2
Figure 2

Safe State: The key to a same being is that there is at least one way for all users to finish. In other analogy, the state of figure 2 is safe because with 2 units left, the banker can delay any request exception C's, thus letting C finish and release all four resource. With four units in hand , the banker can let either D or B have the necessary units and so on. Unsafe State: Consider what would happen if a request from B for one more unit were granted in above figure 2. We would have following situation.

Customer	User	Max
A	1	6
B	2	5
C	2	4
D	4	7

Available Units=1
Figure 3

This is an unsafe state. If all the customers namely A, B, C and D asked for their loans, then banker could not satisfy any of them we would have a deadlock.

Deadlock Detection
Deadlock detection is an important task done by operating system. Deadlock detection came into action because operating system does not take any precautionary measures to avoid the deadlock instead of it detects the deadlock and recover it.

Deadlock detection consists of two cases:
1) Resource has single instance.
2) Resource has multiple instances.

In case of single instance, we make wait- for graph, in which each vertex represents a process and an edge exists between two processes if a resource is owned by latter which is needed by former. Wait-for graph detects the deadlock if a cycle exists.

In case of multiple instance, deadlock detection algorithm is used:
1) Let work be vector of length m. Finnish be vector of length n. Initialize work=Available. For I=0, 1, 2, …..n-1. If Allocation (i)!=0, finish[I]=false, else Finish[I]=true.
2) Find an index such that: finish[I]=true and request[I]<=work. If I does not exists go to step 4.
3) Work=Work+Allocation, finish[I]=true, Go to step 2.
4) If finish[I]=false, for some I, 0<=I<=n
5) The process P(i) is deadlocked, and hence system is deadlocked.

Deadlock Recovery
If deadlock is detected, operating system recovers it by following cases:

1) Kill the process:
In case of processes, either kill the one process which causes the deadlock or kill all the processes involved.

2) Preemption:
In case of resources, preemption allocation of resources is done.

3) Rollback:
With the help of database recorded by operating system, resource is rolled back to n- safe states.

3. Conclusion

Deadlocks are undesirable but unavoidable conditions occurred in the systems due to multiple variables used. The fact of indefinite loop may loosen the confidence of user. The techniques proposed in the paper may give confidence to the user to detect deadlock and recover the system.

References

[1] DIJKSTRA, E. W. "Co-operating sequential processes." In Programming languages: NATO

advanced study institute. F. GENUYS (ED.), Academic Press, London. 1968.

[2] IBM System~360 operating system, supervisor a~*d data m (management services. Form C28-6646- 2, IBM, White Plains, N. Y., 1968.

[3] DENNIS, J. B.; AND VAN HOnN, E. C. "Programming semantics for multiprogrammed computations." Comm. ACM 9, 3 (March 1966), 143-155.

[4] HAVENDEa. J. W. "Avoiding deadlock in nnllti-tasking systems." IBM Systems Journal 2 (1968), 74-84.

[5] BRAUOE, E. J. "An algorithm for the detection of system deadlocks." IBM Technical Report: TROO. 791, IBM Data Systems Division, Poughkeepsie, N. Y., 1961.

[6] DIJKSTaA, E.W. "The structure of the THE multiprogramming system." Comm. ACM 11, 5 (May 1968), 341-346.

[7] HOLT, RICHARD C. "On deadlock in computer systems." (PhD Dissertation) Department of Computer Science. Cornell University, Ithaca, N.Y., Jan. 1971.

[8] P. Brinch Hansen, Operating System Principles, Prentice-Hall, Englewood Cliffs, N. J., 1973, pp. 42–49, 124–125.

[9] N. Habermann, “Prevention of System Deadlocks, ” Comm. ACM, Vol. 12, No. 7, July 1969, pp. 373–377, 385.

[10] R. C. Holt, “Some Deadlock Properties of Computer Systems, ” Computing Surveys, Vol. 4, No. 3, Sept. 1972, pp. 179–196.

[11] W. W. Chu and G. Ohlmacher, “Avoiding Deadlock in Distributed Data Bases, ” Proc. ACM Nat’l Conf., Nov. 1974, pp. 156–160.

[12] N. Chandra, W. G. Howe, and D. P. Karp, “Communication Protocol for Deadlock Detection in Computer Networks, ” IBM Technical Disclosure Bulletin, Vol. 16, No. 10, Mar. 1974, pp. 3471–3481.

[13] S. A. Mahmoud and J. S. Riordon, “Software Controlled Access to Distributed Data Bases, ” INFOR, Vol. 15, No. 1, Feb. 1977, pp. 22–36.

[14] F.J. Maryanski, "A survey of developments in distributed data base management systems, " IEEE Computer, vol. 11, February 1978, 28-38.

[15] R.C. Holt, "Some deadlock properties of computer systems, " Computing Surveys, vol. 4, September 1972, 179-196

[16] Shoshani and A.J. Bernstein, "Synchronization ina parallel accessed data base, " CACM, vol. 12, November 1969, 604-607.

[17] P.F. King and A.J. CoUmeyer, "Database sharing: an efficient mechanism for supporting concurrent processes, " Proc. AFIPS National Computer Conference, 42, June 1973, pp.271-275.

[18] D.D. Chamberlin, R.F. Boyce, and I.L. Traiger, "A deadlock-free scheme for resource locking in a data-base environment, " Information processing 74, Proc. IFIP Congress. Amsterdam: North-Holland Publishing Co., August 1974, pp. 340- 343.

[19] S.A. Mahmoud and J.S. Riordon, "Software controlled access to distributed data bases, " INFOR, vol. 15, February 1977, 22-36.

[20] B. Goldman, "Deadlock detection in computer networks, " Technical Report MIT/LCS/TR-185. Laboratory for Computer Science, M.I.T., Cambridge, Mass., September 1977. (180 pages)

[21] E.G. Coffman, Jr., M.J. Elphick, and A. Shoshani, "System deadlocks, " Computing Surveys, vol. 3, June 1971, 67-78.

[22] G. Le Lann, "Pseudo-dynamic resource allocation in distributed databases, " Proc. Fourth International Conf. on Computer Communications, ICCC-78, Kyoto, Japan, September 1978, pp. 245-251.

Author Profile

Naveen Dubey is a B.tech 3rd year student in department of Computer Science and Engineering College, Bahadurgarh, Haryana, India. His area of interest is Web Development, Machine learning and Deep learning.