Volume 5, Issue 8, August 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Deadlock: A Problem of Computer System

Seema Payal[1], Ruchi Taneja[2]
[1]M.Tech. Scholar, ECE Department, PPIMT, Hisar, India
[2]Asst. Prof., ECE Department, PPIMT, Hisar, 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 systems and, therefore, must be carefully handled by system designers, analysts and engineers. Computer systems are prone to deadlocks. Deadlock is a result of some uncontrolled sequence of release and request of resources 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 detected. . In this paper, we are going to present several algorithms that handle deadlocks in a system. A deadlock can be resolved by aborting one or more processes in the deadlocked-set and restart that process such that its previous state is resumed. A process is aborted when all of the resources it is holding is released, and withdraw all the resource requests it has made.***

Keywords: Deadlock, Conditions for deadlock, Deadlock prevention, Deadlock avoidance, Deadlock detection, Deadlock handling, Deadlock recovery, wait-for-graph, Banker's algorithm.

I. 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 the 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 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.

A process may utilize a resource in the sequence Request, Use and Release. These operations are accomplished by Wait and Signal. A set of processes 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 is detected by the wait-for-graph. Deadlock is removed by different mechanisms like deadlock prevention, deadlock handling and deadlock recovery.

II. EXPLANATION

Necessary deadlock conditions

A deadlock situation can arise if the following four conditions hold simultaneously in a system:

1. Mutual Exclusion Condition

The resources involved are non-shareable. At least one resource (thread) must be held in a non-shareable mode, that is, onlyone process at a time claims exclusive control of the resource. If another process requests that resource, the requesting process must be delayed until the resource has been released.

2. Hold and Wait Condition

Requesting process hold already, resources while waiting for requested resources. There must exist a process that is holding a resource already allocated to it while waiting for additional resource that are currently being held by other processes.

No-Preemptive Condition

Resources already allocated to a process cannot be preempted. Resources cannot be removed from the processes are used to completion or released voluntarily by the process holding it.

3. Circular Wait Condition

The processes in the system form a circular list or chain where each process in the list is waiting for a resource held by the next process in the list.

How to deal with Deadlock?

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

1.The Ostrich Approach

Just ignore the deadlock problem altogether.

2.Deadlock Prevention

Prevent deadlock by resource scheduling so as to negate at least one of the four conditions.

3.Deadlock Avoidance

Avoid deadlock by careful resource scheduling.

4.Deadlock Detection and Recovery

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 condition must hold only for non-sharable resources. That is, several processes cannot simultaneously share a single resource. This condition is difficult to eliminate because some resources, 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 resources. This strategy requires that all of the resources a process will need must be requested at once. The system must grant resources on "all or none" basis. If the complete set of resources needed by a process is not currently available, then the process must wait until the complete set isavailable. While the process waits, however, it may not hold any resources. Thus the "wait for" condition is denied and deadlocks simply cannot occur. This strategy can lead to serious waste of resources.

·Elimination of "No-preemption" Condition

The non-preemption condition can be alleviated by forcing a process waiting for a resource that cannot immediately be allocated to relinquish all of its currently held resources, so that other processes may use them to finish. Suppose a system does allow processes to hold resources while requesting additional resources. 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 resources needed by the first process. This is a deadlock. This strategy require that when a process that is holding some resources is denied a request for additional resources. The process must release its held resources and, if necessary, request them again together with additional resources. Implementation of this strategy denies the "no-preemptive" condition effectively.

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

· 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 resources in order (increasing or decreasing). This strategy impose a total ordering of all resources types, and to require that each process requests resources in a numerical order (increasing or decreasing) of enumeration. With this rule, the resource allocation graph can never have a cycle. .

- **Deadlock Avoidance**

This approach to the deadlock problem anticipates deadlock before it actually occurs. This approach employs an algorithm to access the possibility that deadlock could occur and acting accordingly. This method differs from deadlock prevention, which guarantees that deadlock cannot occur by denying one of the necessary conditions of deadlock.

If the necessary conditions for a deadlock are in place, it is still possible to avoid deadlock by being careful when resources are allocated.

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

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

III. BANKER'S ALGORITHM

This is a deadlock avoidance algorithm. The name 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 no longer satisfy the needs of all its customers. When a new process enters the system, it must declare the maximum number of instances of each resource type that it may need. This number may not exceed the total number of resources in the system. When a user request a set of resources, the system must determine whether the allocation of these resources will leave the system in a safe state. If it will, the resources are allocated; otherwise , the process must wait until some other process releases enough resources.

- In this analogy
  Customers ≡ processes
  Units ≡ resources, say, tape drive
  Banker ≡ Operating System

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

- Available Units=10

Fig.1

we see four customers each of whom has been granted a number of credit units. The banker reserved only 10 units rather 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

Fig. 2

Safe State The key to a state being safe 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 except C's, thus letting C finish and release all four resources. 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

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

Available units= 1

Fig.3

This is an unsafe state. If all the customers namely A, B, C, and D asked for their maximum loans, then banker couldnot satisfy any of them and we would have a deadlock.

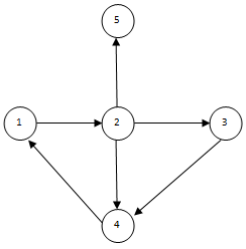
➢ **Deadlock Detection**

Deadlock detection is the process of actually determining that a deadlock exists and identifying the processes and resources involved in the deadlock.

The basic idea is to check allocation against resource availability for all possible allocation sequences to determine if the system is in deadlocked state . Of course, the deadlock detection algorithm is only half of this strategy. Once a deadlock is detected, there needs to be a way to recover several alternatives exists.

Wait-for-graph

If all resources have only a single instance, then we can define a deadlock-detection algorithm that uses a variant called wait-for-graph. We obtain this graph from the resource –allocation graph by removing the nodes of type resource and collapsing the appropriate edges. A deadlock exists in the system if and only if the wait-for graph contains a cycle. To detect deadlocks, the system needs to maintain the wait-for graph and periodically to invoke an algorithm that searches for a cycle in the graph.



- **Recovery from Deadlock**

(a) Process Termination

To eliminate deadlock by aborting a process, we use one of two methods.

- **Abort all deadlocked processes:**

This method will break the deadlock cycle, but at a great expense; these processes may have computed for a long time, and the results of these partial computations must be discarded and probably recomputed.

- **Abort one process at a time until the deadlock cycle is eliminated:**

This method incurs considerable overhead, since, after each process is aborted, a deadlock-detection algorithm must be invoked to determine whether any processes are still blocked.

(b) Resource Preemption

- **Selecting a victim:**

Which resources and which processes are to be pre-empted? As in process termination , we must determine the order of pre-emption to minimize cost. Cost factors may include such parameters as the numbers of resources a deadlock process is holding, and the amount of time a deadlocked process has thus far consumed during its execution.

- **Rollback:**

It is difficult to determine what a safe state is, the simplest solution is a total rollback. Abort the process and then restart it. This method requires the system to keep more information about the state of all the running processes.

- **Starvation:**

How do we ensure that starvation will not occur? In a system where victim selection is based primarily on cost factors, it may happen that the same process is always as a victim. We must ensure that a process can be picked as a victim only a finite number of times.

IV. CONCLUSIONS

It is necessary to instill in users confidence that their application jobs will not be held up at remote locations indefinitely due to deadlock. Without such confidence, there may be a lack of acceptance in a data processing environment, which in turn will impede further progress. The on-line deadlock detection technique proposed in this paper is a step towards increasing that user confidence. If a deadlock is detected, it must be broken by aborting and rolling back at least one process. An effective and efficient rollback and recovery mechanism is of immense need in computer system. Such a mechanism can make our approach to deadlock detection much more attractive. Research efforts are planned in this direction.

REFERENCES

[1] Peterson , J.L. & Silbersehatz “A Operating System concepts”, Addison Wesley.
[2] Brinneh, Hansen “operating system principles: Prentiee Hall of India” .
[3] Haberman, “An introduction to operating” , system design, Galgotia Publication, New Delhi.
[4] Tanenbaum, “An introduction to operating system”.
[5] Hansen, P.B. “Architecture of concurrent programs”,PHI.

