Deadlock handling

Deadlock

System is deadlocked if there is a set of transactions such that every transaction in the set is waiting for another transaction in the set.

- Transaction A places an exclusive lock on item 1001
- Transaction B places an exclusive lock on item 2002
- Transaction A attempts to place an exclusive lock on item 2002
- Transaction A placed into a wait state
- Transaction B attempts to place an exclusive lock on item 1001
- Transaction B placed into a wait state
- This is called a deadlock. One transaction has locked some of the resources and is waiting for locks so it can complete.
- A second transaction has locked those needed items but is awaiting the release of locks the first transaction is holding so it can continue.

Two main ways to deal with deadlock:

a. Prevent it in the first place by giving each transaction exclusive rights to acquire all locks needed before proceeding.

b. Allow the deadlock to occur, and then break it by aborting one of the transactions.

Deadlock Detection

- If the deadlocks are not avoided then another approach is to detect them when they have occurred and recover from them.
- The basic idea is to check allocation against availability for all possible allocation sequences to determine if the system is in deadlocked state.

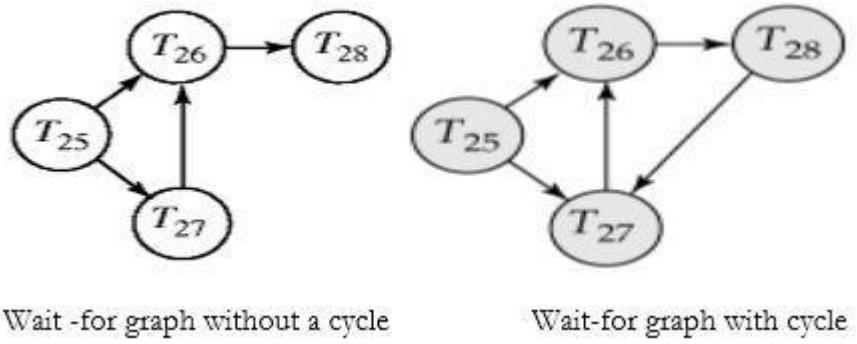
Once detected there needs to be way to recover several alternatives exist:

- Temporarily prevent resources from deadlocked processes
- Back Off a process to some check point allowing preemption of a needed resource and restarting the process at the checkpoint later
- Successively kill processes until the system is deadlock free.

Deadlocks can be described as a wait-for graph, which consists of a pair $G=(V,E)$$G=(V,E)$,

- V is a set of vertices (all the transactions in the system)
- E is a set of edges; each element is an ordered pair $T_i \rightarrow T_j$.$Ti \rightarrow Tj$.

- If $T_i \rightarrow T_j$is$T_i \rightarrow T_j$is in EE, then there is a directed edge from T_iTi to T_jTj, implying that T_iTi is waiting for T_jTj to release a data item. -
  When T_iTi requests a data item currently being held by T_jTj, then the edge $T_i T_j$TiTj is inserted in the wait-for graph. This edge is removed only when T_jTj is no longer holding a data item needed by T_iTi.
- deadlock-detection algorithm periodically to look for cycles.



Wait -for graph without a cycle Wait-for graph with cycle

Deadlock Recovery

Deadlock recovery is generally used when deadlocks are rare and the cost of recovery is low.

Methods for Recovery

1. Termination of processes

- Some victim process is chosen for termination from the cycle of deadlocked processes.
- This process is terminated, requiring a later restart
- All the resources allocated to this processes are released, so that they may be reassigned to other deadlocked processes
- With an appropriately chosen victim process, this should resolve the deadlock.

1. Rolling back processes

- In order to rollback a victim process, there needs to have been some previous checkpoint at which time the state of the victim process was saved to stable storage
- There must also be an assurance that the rolled back process is not holding any resource needed by the other deadlocked processes at that point
- With an appropriately chosen victim process, needed resources will be released and assigned to other deadlocked processes.
- This resolves deadlock.

courtesy

https://www.ques10.com/p/4007/define-deadlock-detection-and-recovery-1/#