Notes of UNIT- 4

Deadlock Detection in Distributed Systems

In this article, you will learn about deadlock detection in the distributed system with its approaches, handling strategies, issues, and resolution.

What is Distributed Deadlock?

Distributed deadlocks can occur when distributed transactions or concurrency control are utilized in distributed systems. It may be identified via a distributed technique like edge chasing or by creating a global wait-for graph (WFG) from local wait-for graphs at a deadlock detector. Phantom deadlocks are identified in a distributed system but do not exist due to internal system delays.

In a distributed system, deadlock cannot be prevented nor avoided because the system is too vast. As a result, only deadlock detection is possible. The following are required for distributed system deadlock detection techniques:

1. Progress

The method may detect all the deadlocks in the system.

2. Safety

The approach must be capable of detecting all system deadlocks.

Approaches to detect deadlock in the distributed system

Various approaches to detect the deadlock in the distributed system are as follows:

1. Centralized Approach

Only one resource is responsible for detecting deadlock in the centralized method, and it is simple and easy to use. Still, the disadvantages include excessive workload on a single node and single-point failure (i.e., the entire system is dependent on one node, and if that node fails, the entire system crashes), making the system less reliable.

2. Hierarchical Approach

In a distributed system, it is the integration of both centralized and distributed approaches to deadlock detection. In this strategy, a single node handles a set of selected nodes or clusters of nodes that are in charge of deadlock detection.

3. Distributed Approach

In the distributed technique, various nodes work to detect deadlocks. There is no single point of failure as the workload is equally spread among all nodes. It also helps to increase the speed of deadlock detection.

Deadlock Handling Strategies

Various deadlock handling strategies in the distributed system are as follows:

1. There are mainly three approaches to handling deadlocks: deadlock prevention, deadlock avoidance, and deadlock detection.
2. Handling deadlock becomes more complex in distributed systems since no site has complete knowledge of the system's present state and every inter-site communication entails a limited and unpredictable latency.
3. The operating system uses the deadlock Avoidance method to determine whether the system is in a safe or unsafe state. The process must inform the operating system of the maximum number of resources, and a process may request to complete its execution.
4. Deadlocks prevention are commonly accomplished by implementing a process to acquire all of the essential resources at the same time before starting execution or by preempting a process that already has the resource.
5. In distributed systems, this method is highly inefficient and impractical.

6. The presence of cyclical wait needs an examination of the status of process resource interactions to detect deadlock.
7. The best way to dealing with deadlocks in distributed systems appears to be deadlock detection.

Issues of Deadlock Detection

Various issues of deadlock detection in the distributed system are as follows:

1. Deadlock detection-based deadlock handling requires addressing two fundamental issues: first, detecting existing deadlocks, and second, resolving detected deadlocks.
2. Detecting deadlocks entails tackling two issues: WFG maintenance and searching the WFG for the presence of cycles.
3. In a distributed system, a cycle may include multiple sites. The search for cycles is highly dependent on the system's WFG as represented across the system.

Resolution of Deadlock Detection

Various resolutions of deadlock detection in the distributed system are as follows:

1. Deadlock resolution includes the braking existing wait-for dependencies in the system WFG.
2. It includes rolling multiple deadlocked processes and giving their resources to the blocked processes in the deadlock so that they may resume execution.

Deadlock detection algorithms in Distributed System

Various deadlock detection algorithms in the distributed system are as follows:

1. **Path-Pushing Algorithms**
2. **Edge-chasing Algorithms**
3. **Diffusing Computations Based Algorithms**
4. **Global State Detection Based Algorithms**

Path-Pushing Algorithms

Path-pushing algorithms detect distributed deadlocks by keeping an explicit global WFG. The main concept is to create a global WFG for each distributed system site. When a site in this class of algorithms performs a deadlock computation, it sends its local WFG to all neighboring sites. The term path-pushing algorithm was led to feature the sending around the paths of global WFG.

Edge-Chasing Algorithms

An edge-chasing method verifies a cycle in a distributed graph structure by sending special messages called probes along the graph's edges. These probing messages are distinct from request and response messages. If a site receives the matching probe that it previously transmitted, it can cancel the formation of the cycle.

Diffusing Computations Based Algorithms

In this algorithm, deadlock detection computation is diffused over the system's WFG. These techniques use echo algorithms to detect deadlocks, and the underlying distributed computation is superimposed on this computation. If this computation fails, the initiator reports a deadlock global state detection.

Global State Detection Based Algorithms

Deadlock detection algorithms based on global state detection take advantage of the following facts:

1. A consistent snapshot of a distributed system may be taken without freezing the underlying computation.
2. If a stable property exists in the system before the snapshot collection starts, it will be preserved.