# Fault Model

DAOS relies on massively distributed single-ported storage. Each target is thus effectively a single point of failure. DAOS achieves availability and durability of both data and metadata by providing redundancy across targets in different fault domains. DAOS internal pool and container metadata are replicated via a robust consensus algorithm. DAOS objects are then safely replicated or erasure-coded by transparently leveraging the DAOS distributed transaction mechanisms internally. The purpose of this section is to provide details on how DAOS achieves fault tolerance and guarantees object resilience.

## Hierarchical Fault Domains

A fault domain is a set of servers sharing the same point of failure and which are thus likely to fail altogether. DAOS assumes that fault domains are hierarchical and do not overlap. The actual hierarchy and fault domain membership must be supplied by an external database used by DAOS to generate the pool map.

Pool metadata are replicated on several nodes from different high-level fault domains for high availability, whereas object data is replicated or erasure-coded over a variable number of fault domains depending on the selected object class.

## Fault Detection

DAOS servers are monitored within a DAOS system through a gossip-based protocol called [SWIM](http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=1028914) that provides accurate, efficient, and scalable server fault detection. Storage attached to each DAOS target is monitored through periodic local health assessment. Whenever a local storage I/O error is returned to the DAOS server, an internal health check procedure will be called automatically. This procedure will make an overall health assessment by analyzing the IO error code and device SMART/Health data. If the result is negative, the target will be marked as faulty, and further I/Os to this target will be rejected and re-routed.

## Fault Isolation

Once detected, the faulty target or servers (effectivelly a set of targets) must be excluded from the pool map. This process is triggered either manually by the administrator or automatically. Upon exclusion, the new version of the pool map is eagerly pushed to all storage targets. At this point, the pool enters a degraded mode that might require extra processing on access (e.g.  reconstructing data out of erasure code). Consequently, DAOS client and storage nodes retry RPC indefinitely until they find an alternative replacement target from the new pool map. At this point, all outstanding communications with the evicted target are aborted, and no further messages should be sent to the target until it is explicitly reintegrated (possibly only after maintenance action).

All storage targets are promptly notified of pool map changes by the pool service. This is not the case for client nodes, which are lazily informed of pool map invalidation each time they communicate with servers. To do so, clients pack in every RPC their current pool map version. Servers reply not only with the current pool map version. Consequently, when a DAOS client experiences RPC timeout, it regularly communicates with the other DAOS target to guarantee that its pool map is always current. Clients will then eventually be informed of the target exclusion and enter into degraded mode.

This mechanism guarantees global node eviction and that all nodes eventually share the same view of target aliveness.

## Fault Recovery

Upon exclusion from the pool map, each target starts the rebuild process automatically to restore data redundancy. First, each target creates a list of local objects impacted by the target exclusion. This is done by scanning a local object table maintained by the underlying storage layer. Then for each impacted object, the location of the new object shard is determined and redundancy of the object restored for the whole history (i.e., snapshots). Once all impacted objects have been rebuilt, the pool map is updated a second time to report the target as failed out. This marks the end of collective rebuild process and the exit from degraded mode for this particular fault. At this point, the pool has fully recovered from the fault and client nodes can now read from the rebuilt object shards.

This rebuild process is executed online while applications continue accessing and updating objects.