Hedvig Deduplication Design User Guide

Table of Contents

Deduplication Design Overview

On the Hedvig platform, the implementation of deduplication is a counter-based approach for primary workloads.

At the backend, a type of virtual disk called a *System Dedup Vdisk* is created to back the user-facing virtual disks. Every System Dedup Vdisk has at most 64 user-facing virtual disks pointing to it. For all of these 64 user-facing virtual disks, only the unique data is written to the System Dedup Vdisk.

Design Details

Creation/deletion workflow

When you create a vdisk and enable deduplication, Hedvig determines (at the backend) if a System Dedup Vdisk already exists, based on the attributes of the user-facing vdisk (for example, Replication Policy, Encryption, Disk Residence, etc.).

If a System Dedup Vdisk does not yet exist, Hedvig creates a new System Dedup Vdisk, using the following naming convention:

HedvigDedup_<AttributesCombination>_Counter_<DiskNum>

Hedvig uses a monotonically increasing counter that keeps track of the number of user disks being created. When the current System Dedup Vdisk has reached its batch size (that is, it has 64 user-facing vdisks mapping to it), a new System Dedup Vdisk is created and made the current one.

When a user-facing vdisk is deleted, the System Dedup Vdisk is not deleted, unless it is the last user-facing vdisk mapped to this System Dedup Vdisk and cannot have any more user-facing vdisk mappings in the future.

Write operation

For any write requests to a user-facing vdisk, Hedvig calculates the md5 hash for each block of data.

The Hedvig Storage Proxy first determines if the same md5 hash exists on its local dedupcache. If the hash is not found, then Hedvig queries the metadata subsystem to see if it exists. If these checks return empty, then Hedvig treats this block as not present for the System Dedup Vdisk and writes this block.

After the write is done successfully, Hedvig updates its local dedupcache for the hash and updates the Metadata subsystem in an asynchronous way. In the future, if the same hash appears again, it will be deduplicated against this System Dedup Vdisk.

On the Hedvig Storage Proxy side, by default, the dedupcache size is decided by dedupcache partition size and memory size. When the threshold is hit, eviction is triggered to clean up the same.

On the server side, the DedupInfo size is decided by `/mnt/f1` size. When the size crosses 90%, new entries are not accepted. In this scenario, the Dedup ratio can be impacted.

Dedup GC Design Overview

On the Hedvig platform, the counter-based System Dedup Vdisk is not deleted until the last user-facing vdisk is deleted. This means that even one remaining user-faced vdisk can block space reclamation for the whole System Dedup Vdisk.

The following sections describe the garbage collection implementation for counter-based dedup vdisks at block level, based on the reference count.

Dedup GC Design Details

Sealing a container

To perform garbage collection for a System Dedup Vdisk, Hedvig first freezes the reference of the blocks, because there is a risk if a block is under garbage collection when a new write request is still trying to point to it.

Hedvig freezes the blocks at container level, which is known as “sealing” the container. Once the container is sealed, none of its blocks will have new write references to them. The number of containers to be sealed is determined by the amount of data that exists for the System Dedup Vdisk and the amount of data that must be kept for all of its user-facing vdisks.

In the background, the server side will run a timer task every 15 minutes to update the sealed containers for a System Dedup Vdisk, and the Hedvig Storage Proxy will fetch the latest sealed containers every 5 minutes.

GC (garbage collection) process

Dedup GC is a background process running on every node. The primary endpoint for the System Dedup Vdisk takes control of the GC for all of the containers of the System Dedup Vdisk.

For every container that is sealed in the GC cycle, all of the user-facing vdisks belonging to this System Dedup Vdisk are listed, and the metadata is queried to determine the blocks that are actually used by these user-facing vdisks. The set of blocks still in use by the user-facing vdisks are treated as “alive” blocks. All other blocks within this container can be garbage collected. The tasks to query metadata for every user-facing vdisk are distributed across the cluster, so they can run simultaneously.

For each sealed container, once the current alive blocks are identified, a GC message is sent out to the container hblock replicas, at the hblock side. This process is called “compaction,” and it triggers this container to throw away the other unused blocks to reclaim space. If there are no alive blocks for this container, then the container, itself, is deleted.

In every Dedup GC cycle, the GC process is re-run on some selected sealed containers from previous cycles. If there has been any garbage generated recently, this space can thus be reclaimed.