Ozone Missing Containers Cleanup

Christos Bisias

Original POC: Ozone Container Cleanup Command

The G-Research Ozone team has run into cases of missing/unrecoverable unhealthy containers, and feel there is insufficient functionality in Ozone to deal with the issue completely.

When a datanode goes down and all containers under it have 0 replicas, then these containers are considered missing and unrecoverable. In that case, the containers and their blocks are inaccessible. If the dead datanode comes back up, missing containers under it, might become healthy again. As long as the datanode is unrecoverable, we end up with missing containers that stay in the system forever and there is no way of cleaning them up.

Hypothetical scenario

There are dead datanodes that can't be recovered. All the data under the datanodes are inaccessible and considered lost for good. The dead datanodes are removed from the cluster.

Recon is reporting a bunch of missing containers and OzoneManager still has metadata for all the keys that belong to the missing containers. The user needs to get rid of all the metadata and Recon's entries of missing containers that generate all these reports.

Proposed approach

There should be an

`containerCleanup` command for cleaning up missing containers. This command is only to be used as a last resort to restore the cluster to its original state.

The user has performed all actions possible to restore the datanodes and has decided that at this point the datanodes and their data are not salvageable. The datanodes are removed from the cluster and their data are considered lost for good. The user needs a way to clean up Recon, SCM and OM. That’s the only case where the above command should be used.

CLI command

The command will accept a container-id and might be later extended to accept datanode-uuid, datanode-host and pipeline-id.

For a provided container-id, the command will validate that the ID belongs to a missing container and that the container still exists in the SCM and then it will proceed and enter the ID to a RocksDB table.

A background service will pick the ID from the table and clean up the container.

Background Service

The container cleanup operation is heavy and time consuming and therefore will be performed asynchronously by a periodic background service.

The background service will pick an ID from the table and follow the next steps

- Get all container replicas from the SCM, to verify that the container is missing
    - There should be no replicas, the list should be empty
- Get all keys from the Recon that are associated with the provided ID
- Delete all keys from the OM
- Send a DeleteContainer request to the SCM
    - `StorageContainerLocationProtocol.deleteContainer()`
    - On the SCM, all the methods for deleting a container already exist but `DeleteContainer` is not a recognizable command. We have to expose it.

Key Deletion

Recon keeps track of all keys associated with a container and makes the information easily accessible. We will use Recon to get all the container keys.

The normal key deletion service wouldn't be efficient in this case. For missing containers, all their blocks are inaccessible and trying to delete them would fail. Key deletion for a missing container will be different from the regular operation and will be performed as part of the container cleanup service.

After getting all the keys from Recon, we should check the FileTable and KeyTable and delete the keys. We can't delete blocks that belong to missing containers as the operation will fail in the SCM with a ContainerNotFoundException. A key can be large enough to spread across multiple containers. We should also consider dealing with blocks that might be left orphaned under non-missing containers.

Cleaning up Recon

Recon gets all of its container updates from the datanodes and then updates its own version of SCM.

The SCM doesn't communicate directly with Recon or send any updates. Normally, if a container was deleted, the datanode would report the change to Recon. In this case, the datanode that holds the container is dead and inaccessible.

We can modify and use Recon's periodic health task which checks for missing and unhealthy containers and then verifies if they still exist in the SCM. The health task will check the SCM and if the container has been deleted there, it will remove it from the unhealthy containers table and then use `ReconContainerManager` to delete it.

`ReconContainerManager.deleteContainer()` will take the following actions

- Remove the container from Recon's SCM container state map
- Delete the container from recon-scm.db containers table
- Delete the container from all the mapping tables
    - containerKeyCountTable
    - containerKeyTable
    - keyContainerTable

Keep the container record in containerReplicaHistoryTable.

Recon's container health check runs every 5 minutes by default. We don't have to do anything but wait for the next sweep.

Dead datanode becomes healthy again

There is a chance that the dead datanode which was considered unrecoverable, might become healthy again. In case the user has run the container cleanup command, all containers under the datanode will have been deleted from SCM and therefore will be considered unknown.

Flag `hdds.scm.unknown-container.action` can be used to specify what will happen to the containers. The default value is `WARN` but if set to `DELETE`, then all the unknown containers will be cleaned up from the system. `WARN` keeps logging that there is an unknown container but takes no action on it.

A follow up task can be to create another container command `ozone admin container restore` which will process the container and its blocks and store all the necessary information on the OM and the SCM, so that the container might be

recognizable again by the system.

US Community Sync discussion

Below are the main points from the community meeting discussion on 22 May 2023.

- Check why objects are missing
- Check how many objects are missing
- What's the effect of deleting all these missing objects
- Scanning the OM tables in the background will affect performance
- Background scanning should happen in Recon
- Recon's key mapping should be used instead of scanning OM's tables
- Recon can be unreliable
- A container is safe to be deleted only if it has no key mappings
- Recon should move all the missing container data to new tables and hide them from the UI, so that the user can still review the data
- Recon can have a new tab with all the missing container data
- Do we want to recover blocks from non-missing containers when deleting keys?
- Normal key deletion can’t be reused because accessing missing container blocks on the SCM results in an error
- Recon is not HA, where do we store deleted keys?