Swift object replicator today

=============================

How does swift store objects (\*.data) on disk?

-----------------------------------------------

Background on terminology:

Assume we upload object named "myobject" into container "mycontainer" inside of the account "myaccount".

Swift now determines where to place the object by calculating the

hash of the object = md5('myaccount/mycontainer/myobject') to determine the following:

- object ring's partition number into which the object belongs

- device(s) on which object needs to be plcaed

- object's suffix directory

- object's hash directory

- finally, the object is stored with the timestamp as its name with an extension of .data

After obtaining the above information, swift will then calculate hashes.pkl for each of the ring partition directory.

'hashes.pkl' simply contains the hash for each suffix directory (a9c) inside the given partition directory.

swift-object-replicator process in each of its run, attempts to compare hashes.pkl across all the primary nodes that

belong to that partition. Once a mismatch is detected, the entire suffix directory's contents are rysnc'ed and updated

based on the timestamps. When an object is deleted or an object expires, the object-auditor process will list that

specific object's hash in the hashes.invalid file.

/srv

└── node

├── sdb3

│   ├── objects

│   │   ├── 847 ----------------------------------------> ring partition directory

│   │   │   ├── a9c ------------------------------------> suffix directory

│   │   │   │   └── d3f2d3bfcb4f66177f93384e0756ca9c ---> hash (of the object name) directory

│   │   │   │   └── 1487355585.12092.data ---> object file stored with timestamp as its name

│   │   │   ├── hashes.invalid -------------------------> list of invalidated hashes per partition

│   │   │   └── hashes.pkl -----------------------------> pickled object of hashes of contents for each

| | suffix directory per partition

│   ├── objects-1

│   ├── objects-2

│   └── tmp

10 directories, 3 files

Now assume, we delete the object file manually, i.e.

sudo rm /srv/node/sdb3/objects/847/a9c/d3f2d3bfcb4f66177f93384e0756ca9c/1487355585.12092.data

and expect swift object replicator to handle this .data file recovery for us? It is not!

What swift-object-replicator does today?

-----------------------------------------

Today's implementation of swift object replicator, runs through the ring partition directories, to check the

validity of "hashes.pkl" file. If hashes.pkl seems to be modified, it then invalidates the

hashes (inside of hashes.pkl pickled dictionary) for suffix directories modified, and rsyncs the entire sufix directory

contents to the remote servers.

Swift is designed to handle all the data changes that happen through filesystem or pagecache only.

Manually deleting .data file is treated as using an O\_DIRECT flag to write to the block device. This kind of intervention

to filesystem is beyond the purvue of Swift.

With any other kind of operations such as POST, DELETE, EXPIRE-AT, we always go through the filesystem and invalidate the

suffix thereby marking it for replication.

What is the problem to solve?

-----------------------------

At a very high level, the frequency at which hashes.pkl file needs to be calculated must be determined. In the above scenario,

when we manually deleted the .data file, if we manually recalculated and updated the hashes.pkl file, everything would have

worked normally because replcaitor in its next run, will stumble upon the changed hashes.pkl and will recover the cluster by

comparing the time stamps across the respective primaries. However, that is not the case with manual deletion of .data file.

Hence we need to find a way to timely recalculate the hashes.pkl to keep the cluster consistent and durable.

References:

-----------

https://github.com/openstack/swift/blob/0349949030a434fc968da2021ba89000e2cc0152/swift/obj/diskfile.py#L327

https://docs.openstack.org/developer/swift/overview\_replication.html

<https://docs.openstack.org/developer/swift/overview_ring.html>

