Making RBD snapshot based mirroring robust for disaster recovery

Ramana Raja
IBM

Introduction to RBD mirroring

Asynchronous replication of images between clusters carried out by rbd-mirror daemon

Two modes of replication:

- **Journal-based**
  - Writes to journal before writing to primary image (2X write latency)
  - Mirror daemon reads from journal and replays changes on the non-primary image
- **Snapshot-based (focus of talk)**
  - Schedule crash-consistent mirror-image snapshots on primary image
  - Mirror daemon identifies the data/metadata changes between mirror-snapshots
  - Mirror daemon copies the snapshot delta to the non-primary image

To note:

- Enabling ‘fast-diff’ helps determine updated data blocks quickly without scanning full image
- If ‘fast-diff’ not enabled, mirroring will work but will be slower
- Partially applied delta rolled back during failover

Introduction to RBD mirroring

RBD mirroring supports two different configurations

- **One-way replication**
  - Data replicated from primary cluster to a secondary cluster
  - Mirror daemon runs only on secondary cluster
- **Two-way replication**
  - Data replicated from primary images on one cluster to non-primary on second cluster (and vice-versa)
  - Mirror-daemon runs on both clusters

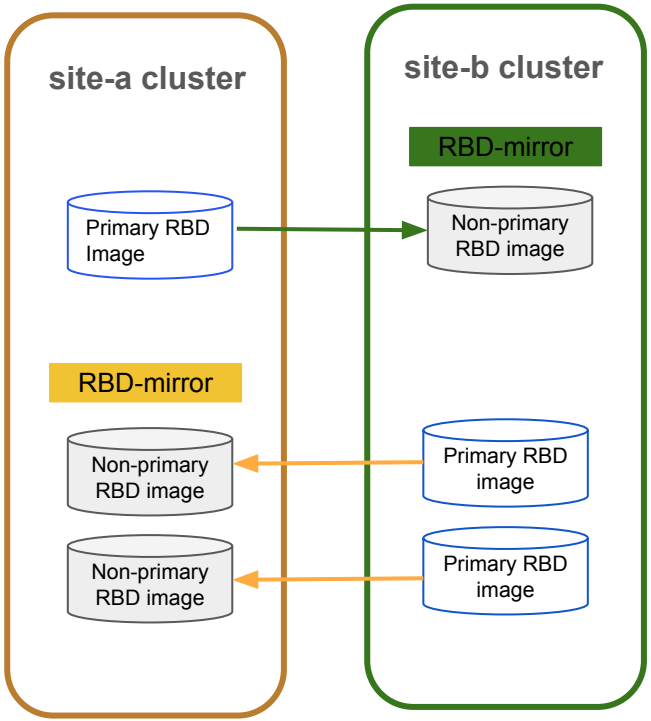


Fig: Two-way replication

Snapshot based mirroring setup

- **Enable mirroring on pool**

  rbd mirror **pool enable** [--site-name {local-site-name}] {pool-name} **image**

  - Unlike with journal-based mirroring, snapshot-based mirroring must be explicitly enabled on each image

- **Bootstrap peers**

  rbd mirror pool **peer bootstrap create** [--site-name {local-site-name}] {pool-name}

  rbd mirror pool **peer bootstrap import** [--site-name {local-site-name}]
  [--direction {rx-only or rx-tx}] {pool-name} {token-path}

  - Registers peer and creates user account for mirror-daemon to discover peer cluster

Snapshot based mirroring setup

- **Enable mirroring on image**

  rbd mirror **image enable** {pool-name}/{image-name} **snapshot**

- **Create mirror snapshots of image**

  rbd mirror **image snapshot** {pool-name}/{image-name}

  - Recommended: schedule automatic creation of mirror-snapshots using ‘rbd_support’ ceph-mgr module

    rbd mirror **snapshot schedule add** [--pool {pool-name}] [--image {image-name}] {interval} [{start-time}]

Mirroring primitives for planned failover

- **Demote primary image**

  rbd mirror **image demote** {pool-name}/{image-name}

  - Marks the image as non-primary (unwriteable to standard RBD clients)
  - Creates a demote snapshot

- **Promote non-primary image**

  rbd mirror **image promote** {pool-name}/{image-name}

  - Works only after demote snapshot is fully synced
  - Creates a promote snapshot
  - Marks the image as primary (writeable to standard RBD clients)

Mirroring primitives for unplanned failover

- **Force promote image**

  rbd mirror **image promote** –force {pool-name}/{image-name}

  - If changes not fully synced, rolls back image
  - Creates a promote snapshot
  - Marks the image as primary (writable to standard RBD clients)

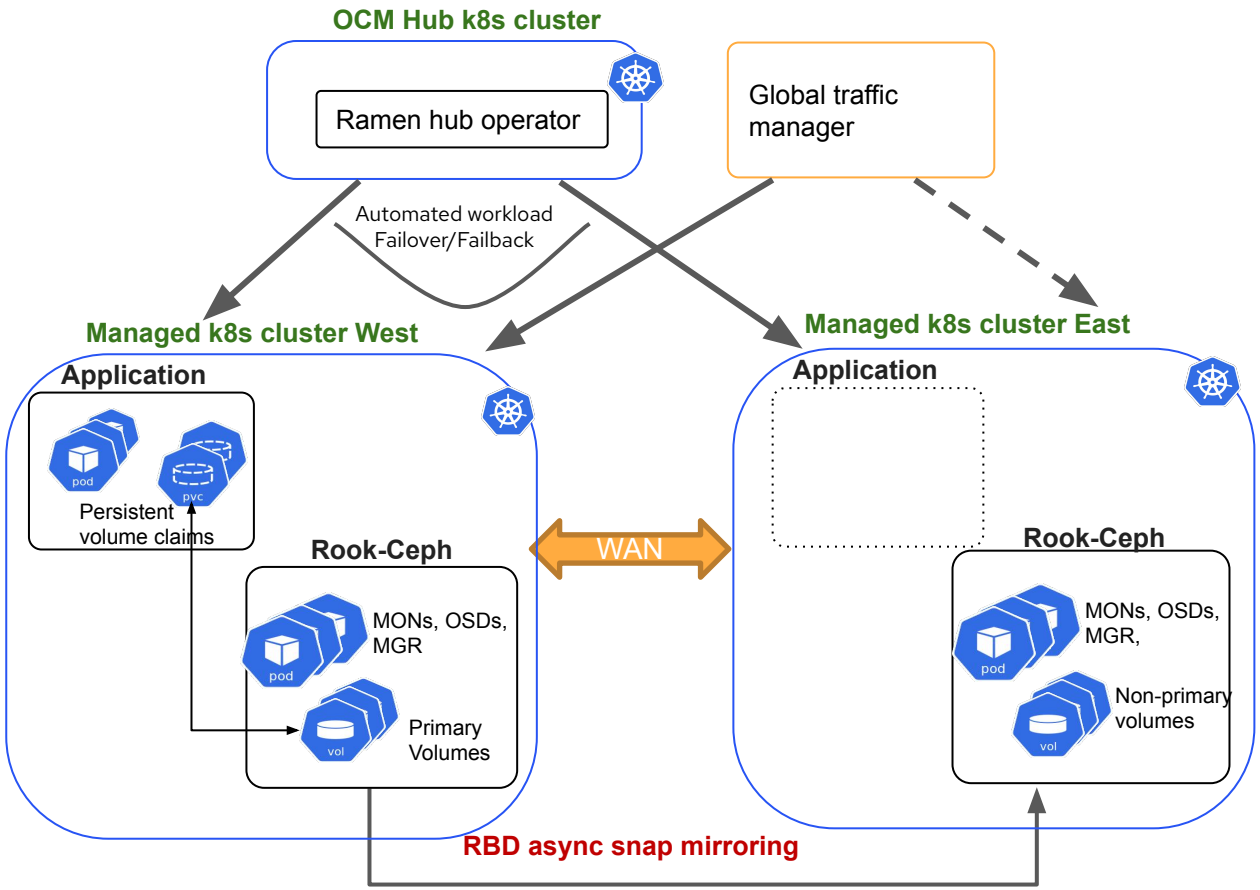
  Note:

  - Force promotion leads to split-brain between peers. Split brain will be detected by mirror-daemon
  - Split brain resolved by demoting the out-of-date image and then requesting resync
- **Force resync image**

  rbd mirror **image resync** {pool-name}/{image-name}

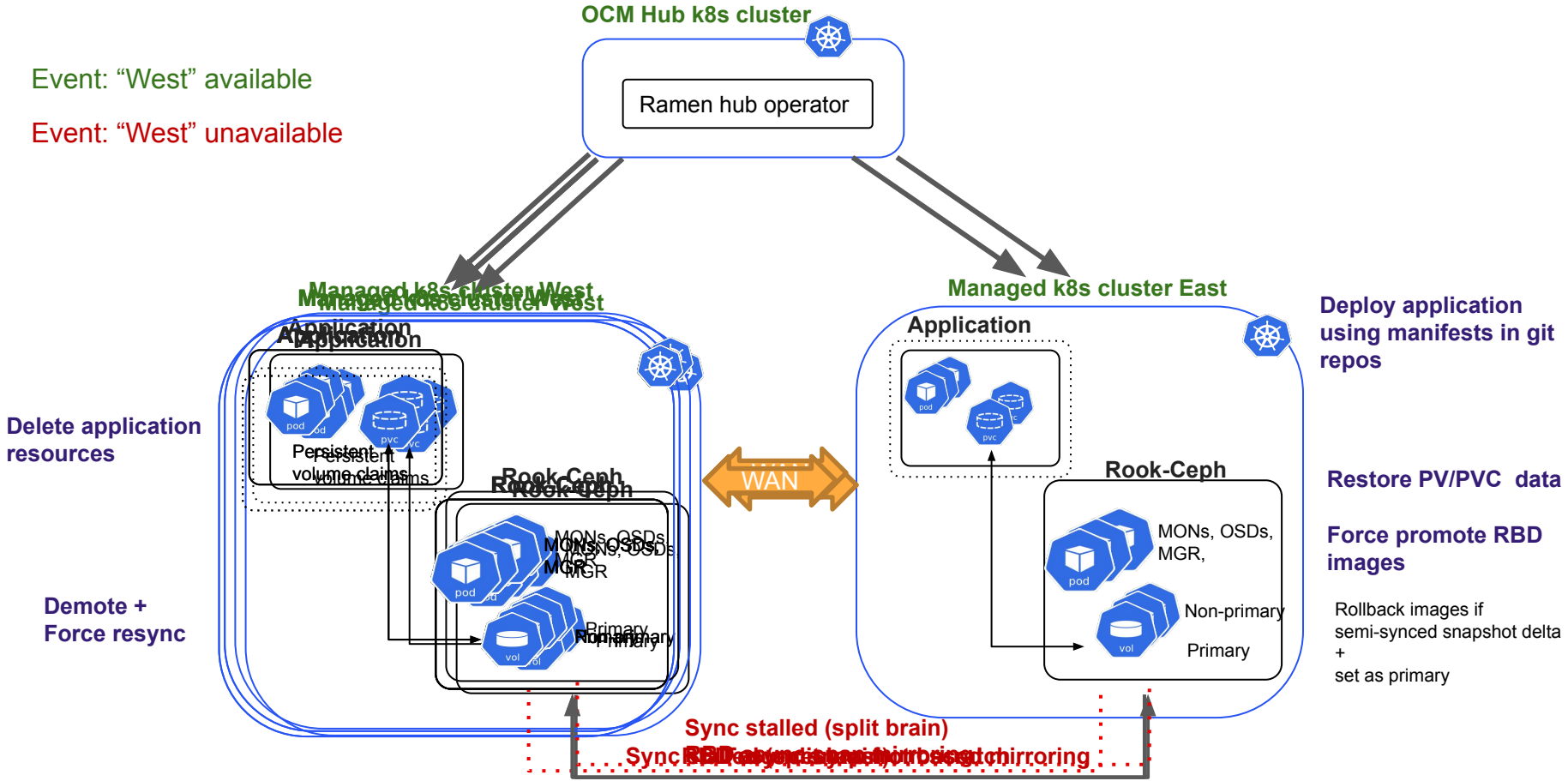
  - Deletes the demoted image and then resyncs from primary
  - Image resynced from scratch

Regional disaster recovery architecture in kubernetes

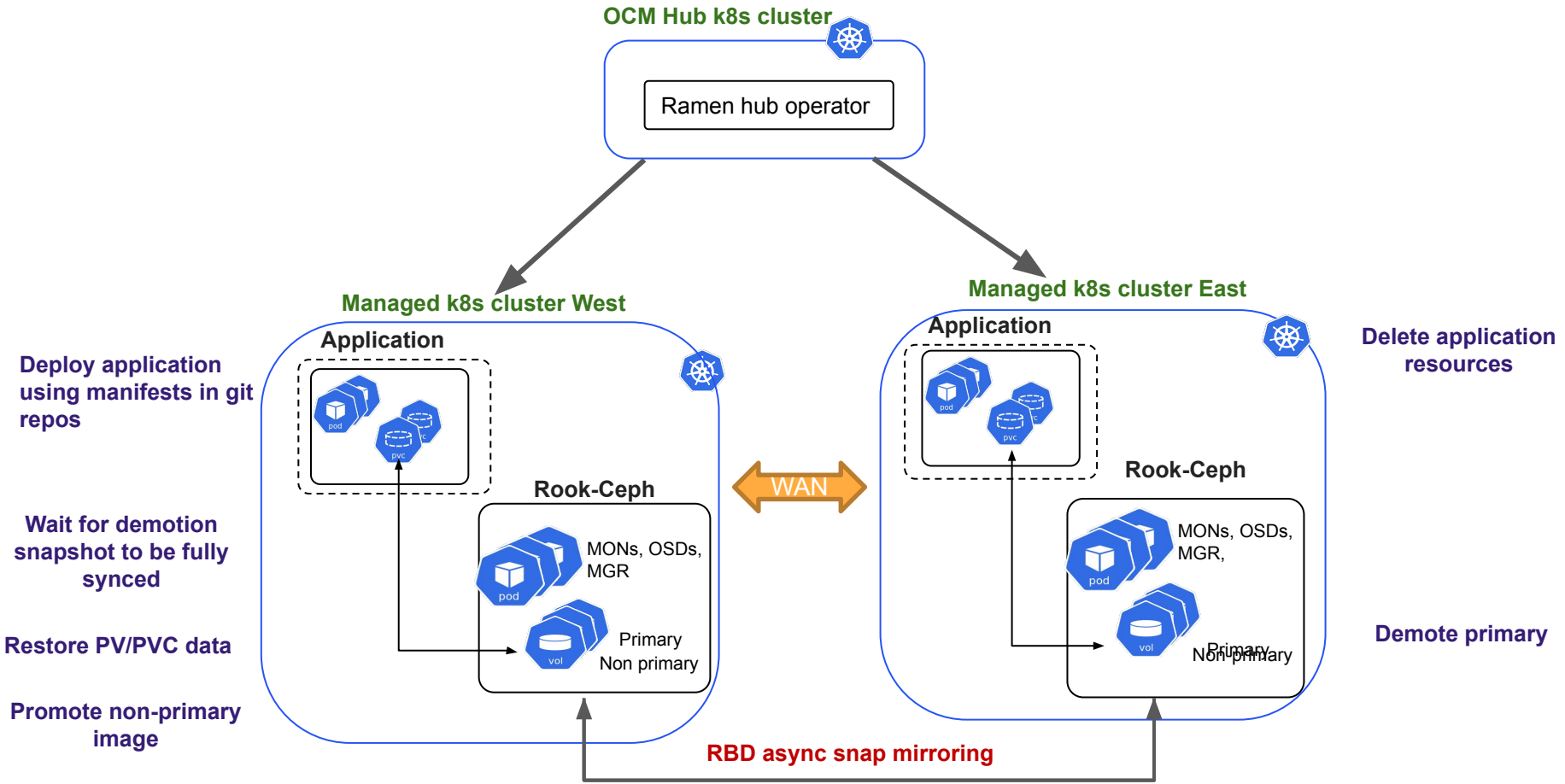


- Application recovery during data center outages in an entire region
- Network latency > 10 ms
  Recovery time objective in mins
  Recovery point objective in mins
- 3 cluster solution in Open Cluster manager (OCM) platform
  (hub cluster + 2 managed clusters)
- Ramen operator orchestrates placement of application and its storage
- Persistent volumes backed by RBD images asynchronously replicated using snapshot-based RBD mirroring

Failover Orchestration



Failback/Relocate Orchestration



Volume data corruption during failover

Issue

- Mirror snapshots taken during active I/O (e.g., untar workload)
- Planned failover (demote primary + promote non-primary)
- Promoted image corrupted. Observed file system and block level corruption

Root cause

- Object maps of mirror snapshots didn't reflect actual contents of snapshots
- Synced incorrect snapshot delta, which was calculated based on corrupted object map

Fixes

- librbd and krbd: avoid object map corruption in snapshots taken under I/O
  https://github.com/ceph/ceph/pull/52109
  https://github.com/ceph/ceph-client/commit/870611e487
- Added functional tests to validate that images under I/O are mirrored correctly
  ceph/qa/workunits/rbd/compare_mirror_images.sh
  ceph/qa/workunits/rbd/compare_mirror_image_alternate_primary.sh

Mirror snapshot scheduler and blocklisting

Issue

- In Ceph clusters with constrained resources in k8s environment, snapshot scheduler occasionally stopped working; mirror snapshots not created
- Snapshot-scheduler part of ‘rbd_support’ ceph-mgr module
- ceph-mgr had to be restarted affecting other mgr services (not okay)

Root cause

- Snapshot scheduler’s client was blocklisted by kernel RBD clients wanting exclusive locks

Fixes

- krbd: fixed issue with erroneous blocklisting of other clients
  https://github.com/ceph/ceph-client/commit/588159009d
- ceph-mgr: made snapshot-scheduler (rbd_support module) recover from blocklisting
  https://github.com/ceph/ceph/pull/49742
- librbd: fixed ExclusiveLock state machine to propagate blocklist error to caller
  https://github.com/ceph/ceph/pull/53829

Possible hangs in mirroring under high latency

Issue

- Observed unexpected slow down in rbd-mirror daemon mirroring images

Root cause

- rbd-mirror daemon in non-primary cluster tried to remove old mirror snapshots of primary image
- If latency between was high enough, mirror daemon couldn’t acquire exclusive lock of primary image in time and kept retrying

Fixes

- librbd: localize snap-remove operation of synced old mirror snapshots on primary cluster
  https://github.com/ceph/ceph/pull/51166/
- librbd: clean up demotion snapshots explicitly
  https://github.com/ceph/ceph/pull/53251

Ongoing work

- Make snapshot based mirroring of clones work
  - https://github.com/ceph/ceph/pull/55892
- Make snapshot based mirroring propagate discards to secondary
  - https://github.com/ceph/ceph/pull/52358
- Supporting snapshot based mirroring of image groups
  - https://github.com/ceph/ceph/pull/53793

Future Work

- Rigorous testing of multiple RBD mirror daemons load balancing the syncing of multiple images
- More efficient than force-resyncing from scratch

Takeaways

- Regional disaster recovery of kubernetes container workloads using RBD storage
  - RBD mirroring setup:
    - Two-way snapshot-based mirroring
    - Single RBD mirror daemon on each cluster
  - failover/failback/relocate of application and its storage orchestrated by Ramen operator

- Lots of improvement in various layers of RBD snapshot based mirroring feature
  - librbd client
  - kernel rbd client
  - RBD mirror-snapshot scheduler in mgr-module

- Upcoming features/fixes:
  - Mirroring of image groups
  - Mirroring of RBD clones