SUSE® Enterprise Storage: Disaster Recovery with Block Mirroring

Ricardo Dias
Senior Software Engineer
SUSE Linux / rdias@suse.com

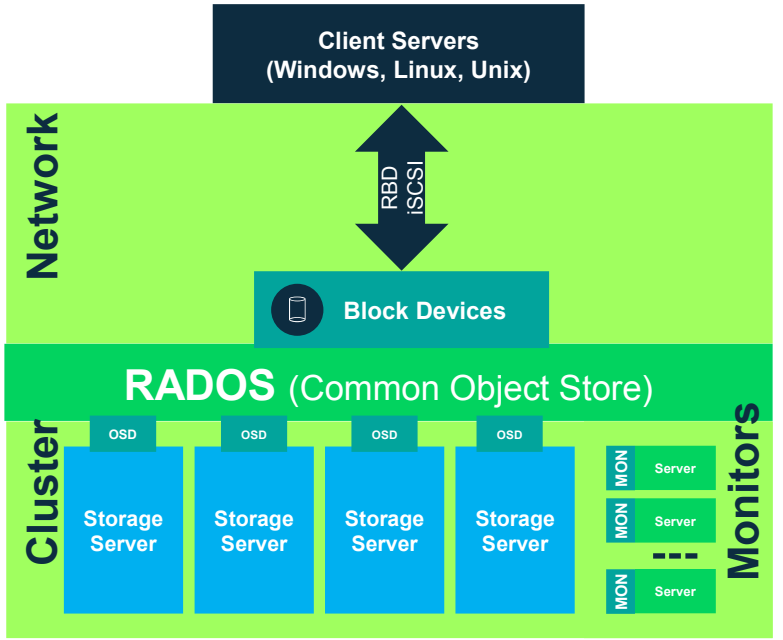
Motivation

Motivation

- **Data is an important asset**
- **Data size growth requires a storage system that is:**
  - Highly scalable;
  - Cost effective;
  - and, Fast
- **SUSE Enterprise Storage (powered by ceph)**
- **Highly resilient and self-managed storage solution**
- **Unified system that exposes several interfaces:**
  - Object: RADOS (librados), and RGW (RADOS Gateway)
  - Block: RBD (RADOS Block Device)
  - File: CephFS

RADOS Block Device

- **Block device abstraction**
- **Thin provisioned**
- **Resizable**
- **Snapshots and clones**
- **Several clients:**
  - libRBD
  - Kernel module
  - iSCSI
  - QEMU



Ceph and Fault Tolerance

- Distributed storage system
- Data is replicated in a single cluster
- Strong consistency per object (no transactions)
- Tolerates faults up to N-1 nodes (where N is the replication factor)

- But in the real world… things like these can happen:

Disaster Recovery

- Automatic Backup
- Distant place
- Easy manageable
- Invisible (to the users)
- (Semi-)Automatic Recovery

What can Ceph do for us?

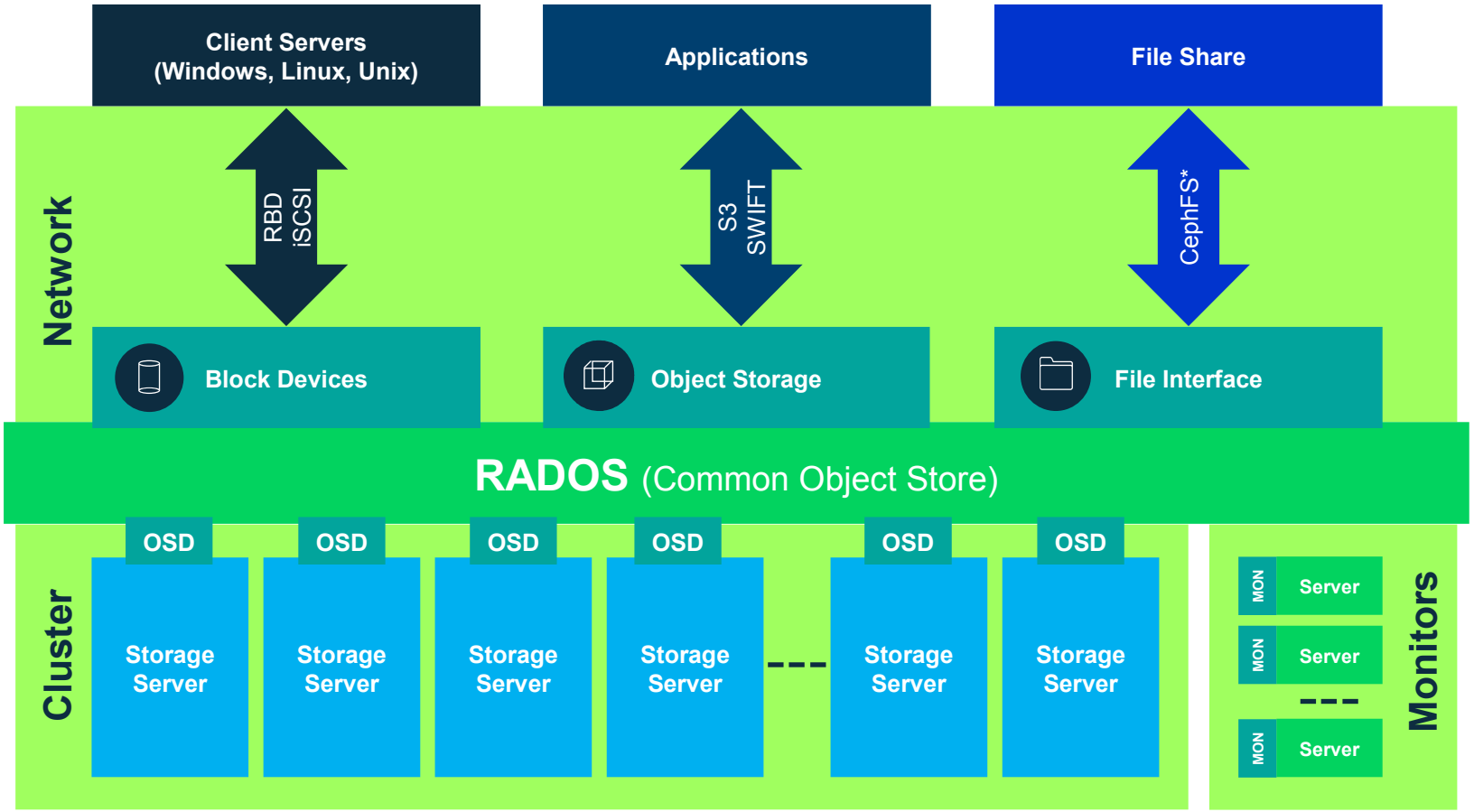
Outline

- **Brief introduction to Ceph**
- **RADOS Block Device (RBD)**
- **RBD Mirroring**
  - Architecture
  - Journaling
  - Internals “RBD-Mirror daemon”
  - Failure handling
  - Management
- **Future features**

Brief introduction to Ceph

Ceph distributed storage

- **Core component of SUSE Enterprise Storage**
- **Highly resilient and self-managed storage solution**
- **Unified system that exposes several interfaces:**
  - Object: RADOS (librados), and RADOS gateway (RESTfull interface)
  - Block: RBD (RADOS Block Device)
  - File: CephFS
- **Data stored in hundreds of nodes**
  - Distribution is controlled by the CRUSH algorithm
  - Horizontal scalability


Client Servers
(Windows, Linux, Unix)
Applications
File Share
Network
RBD
iSCSI
S3
SWIFT
CephFS*
Block Devices
Object Storage
File Interface
RADOS (Common Object Store)
Cluster
OSD
Storage Server
Monitors
MON
Server

RADOS Block Device (RBD)

RADOS Block Device (RBD)

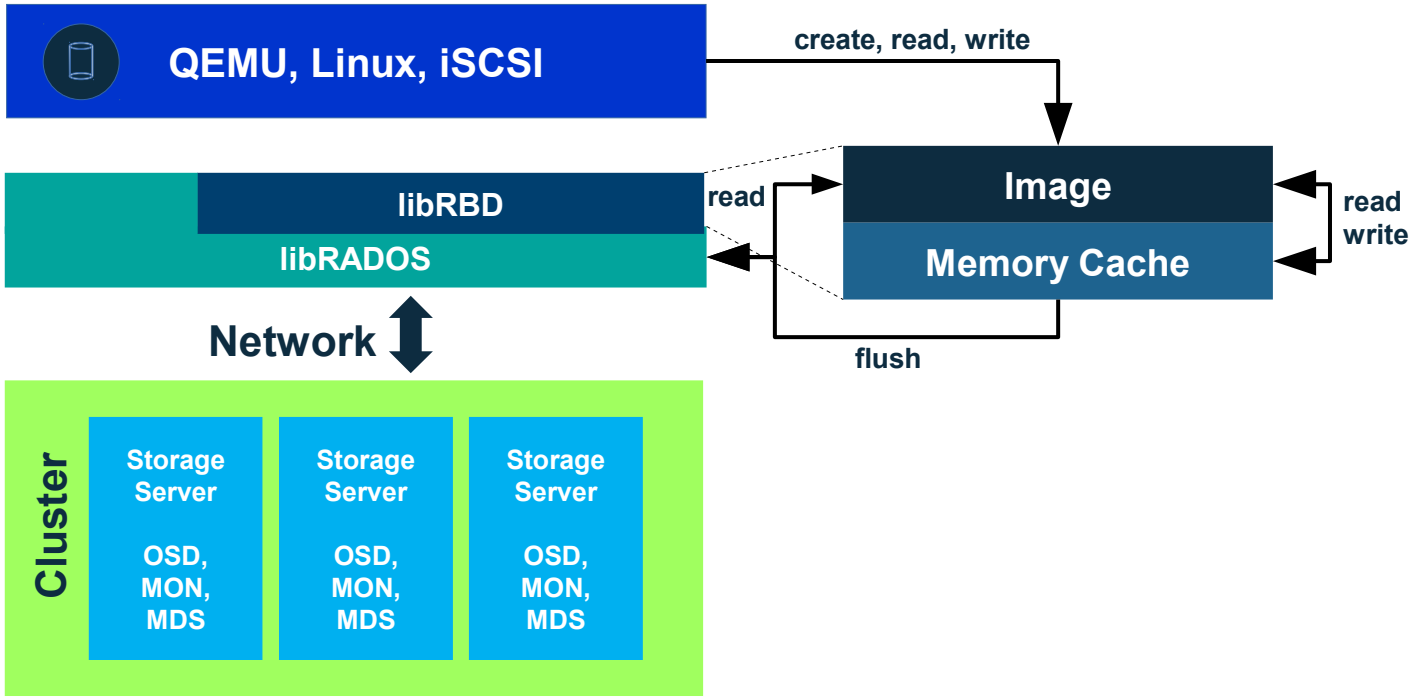
- **Block device abstraction**
- **Can be used as:**
  - Mount point (with kernel module)
  - VM virtual volumes (QEMU librbd bidings)
  - iSCSI
- **Block device is called “RBD Image”**
- **Thin provisioning**
- **Images are striped in objects of fixed size**
  - Stored in OSDs

RBD Images

- **You know nothing OSD!**
  - OSDs are oblivious to RBD images
- **RBD client library “libRBD”**
  - Maps image operations to OSD operations
  - Memory cache

- **Image abstraction only exists in the client side**

RBD Architecture*



* over-simplification of the architecture

Back to the start!

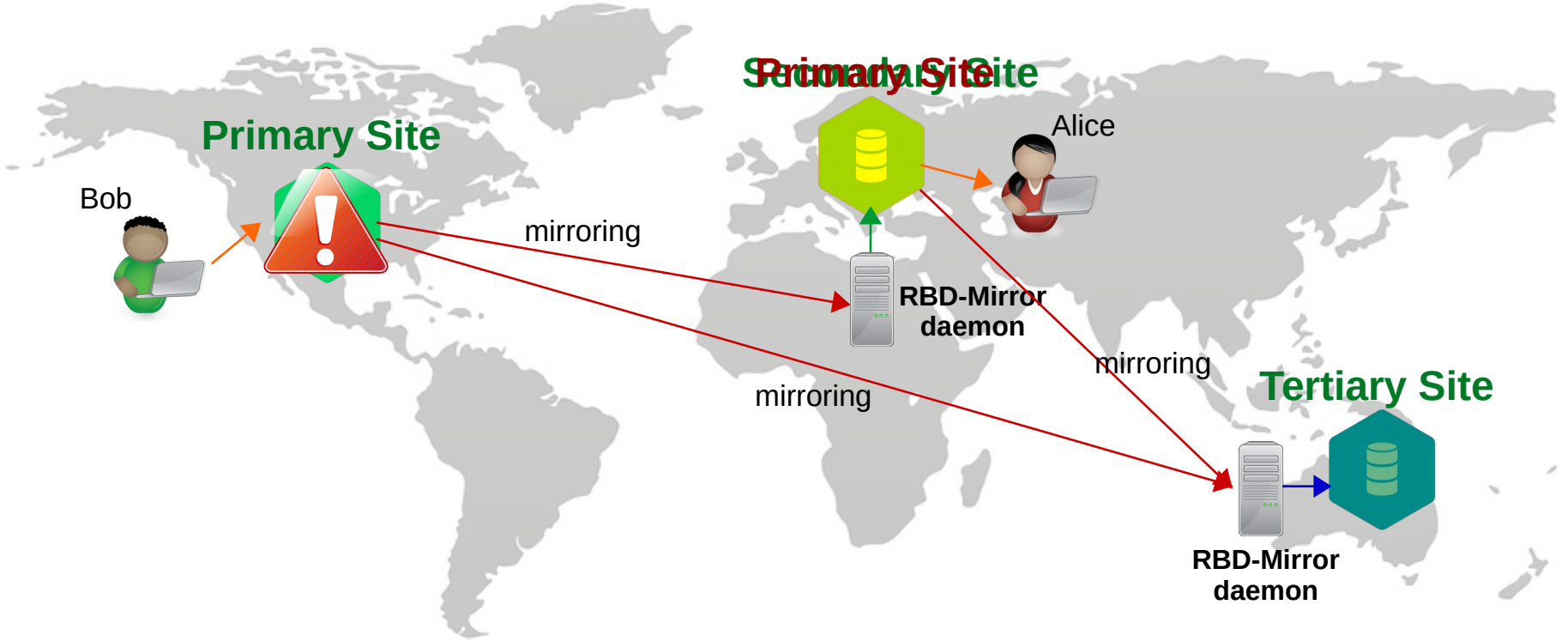
- **BACKUPS!**
- **What can Ceph do for us?**

- **RBD images are stored in a single cluster**
  - A cluster is composed by a set of servers
  - Interconnected by a LAN
  - Single data center

- **Asynchronously mirror RBD images to a remote site**
  - RBD Mirroring
  - Since Ceph **Jewel** release
  - Technical Preview in **SES3** and **SES4**

RBD Mirroring

RBD Mirroring



RBD Mirroring

- **Asynchronous replication**
  - rbd-mirror daemon fetches data from the primary site
  - rbd-mirror daemon writes data in local site
- **rbd-mirror daemon requires access to both remote and local clusters**
- **Crash Consistency**
  - Write events and barriers are preserved during replication

RBD Mirroring - Journaling

- **Mirroring relies on the new RBD image feature: Journaling**
- **Image Journal:**
  - Logs all operations that change the contents and structure
    - Writes
    - Resizes
    - Flushes
    - Etc…
  - Sequential
  - Stored as objects in OSDs

Journaling

- **How are operations logged to the journal?**
  - libRBD takes care of this, example:
    1. client: issues a write operation onto an image
    2. librbd: generates a journal order number and tags the write event
    3. librbd: sends the tagged write event to the OSD asynchronously
    4. librbd: writes the value to the memory cache
    5. librbd: returns to the client
  - Cache entries can only be evicted after receiving the journal commit
- **rbd-mirror daemon reads the events from the image journal on the primary site**
- **Then, replays the same events in the local site**

RBD-Mirror Daemon

- **The core component of rbd-mirror feature**
- **Runs in one of the Ceph cluster servers**
- **Requires access to the primary site monitors and OSDs**
- **Performs all mirroring work**
  - Reads journal entries in the primary site
  - Replays entries in the local cluster
- **Syncs image upon bootstrap**
  - Also, resyncs
- **Handles fail-over operations**
  - Promotions, Demotions

Primary Site Failure

- **When the primary site fails:**
  - demote the primary image to non-primary
  - promote the secondary image to primary
    - Force promote, if primary site is completely inaccessible
- **Re-sync of images**
  - When failed site becomes accessible again

Managing RBD-Mirror

Prepare Image for Mirroring
(Primary Cluster)

Assumptions:
Pool name: `rbd`
Images: fooimg

- **Enable journal feature**
  - rbd -p rbd feature enable journaling fooimg journaling
- **Enable Pool Mirroring Mode**
  - rbd -p rbd mirror pool pool
    - Mirroring is enabled for all images that have journaling enabled
  - rbd -p rbd mirror pool image
    - Pool mirroring mode is enabled but no image mirroring is enabled
- **Enable Image Mirroring Mode**
  - rbd -p rbd mirror image enable fooimg
    - Only works if pool is in "**image**" mode

Mirror Cluster Configuration
(Secondary Cluster)

Assumptions:
Pool name: `rbd`
Primary cluster: `primary_cluster`

- **Enable Pool Mirroring Mode**
  - rbd -p rbd mirror pool pool
    - Mirroring is enabled for all images that have journaling enabled
- **Add Peer info**
  - rbd -p rbd mirror pool peer add primary_cluster
    - `primary_cluster.con` ust exist

rbd-mirror daemon is watching peer information, and starts mirroring automatically

Fail-over procedure

Assumptions:
Pool name: `rbd`
Images: fooimg

- **Demote image in Primary Cluster**
  - rbd -p rbd mirror image demote fooimg
- **Promote image in Secondary Cluster**
  - rbd -p rbd mirror image promote fooimg

Future Features

RBD-Mirror Future Features

- **RBD-Mirror Daemon High Availability**
  - Currently only one daemon is mirroring – No fault tolerance
  - Allow multiple daemons to co-exist
    - Fault-tolerance
    - Load balancing

- **Consistency Groups**
  - Crash consistency for multiple RBD images
  - Group images share the journal

Questions?


SUSE®
We adapt. You succeed.