Disaster Recovery and Ceph Block Storage

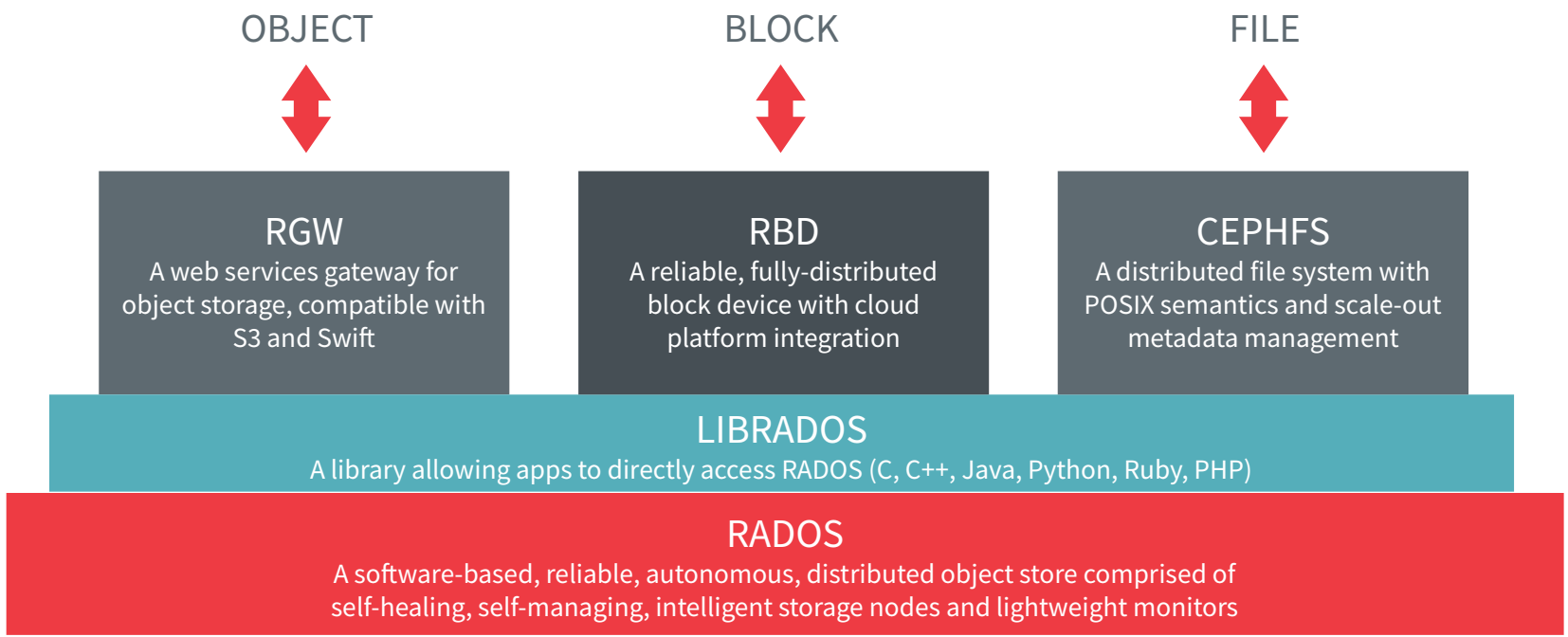
Introducing Multi-Site Mirroring

Jason Dillaman
RBD Project Technical Lead
Vault 2017

WHAT IS CEPH ALL ABOUT

- Software-defined distributed storage
- All components scale horizontally
- No single point of failure
- Self-manage whenever possible
- Hardware agnostic, commodity hardware
- Object, block, and file in a single cluster
- Open source (LGPL)

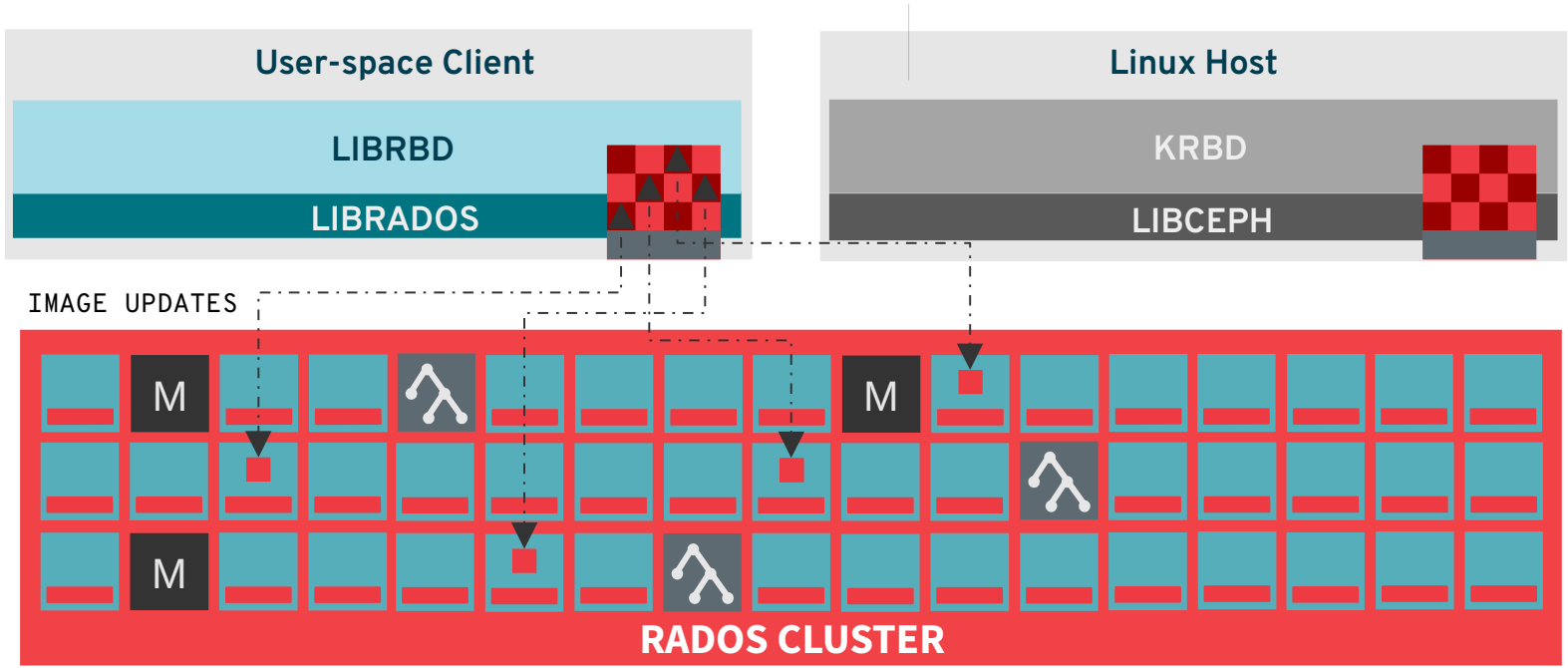
CEPH COMPONENTS



RADOS BLOCK DEVICE

- Block device abstraction
- Striped over fixed-size objects
- Highlights
  - Broad integration
  - Thinly provisioned
  - Snapshots
  - Copy-on-write clones

RADOS BLOCK DEVICE



MIRRORING MOTIVATION

- Massively scalable and fault tolerant design
- What about data center failures?
  - Data is the “special sauce”
  - Failure to plan is planning to fail
- Snapshot-based incremental backups
- Desire online, continuous backups
  - a la RBD Mirroring

MIRRORING DESIGN PRINCIPLES

- Replication needs to be asynchronous
  - IO shouldn’t be blocked due to slow WAN
  - Support transient connectivity issues
- Replication needs to be crash consistent
  - Respect write barriers
- Easy management
- Expect failure can happen anytime

MIRRORING DESIGN FOUNDATION

- Journal-based approach to log all modifications
- Support access by multiple clients
- Client-side operation
- Event logs appended into journal objects
- Delay all image modifications until event safe
- Commit journal events once image modification safe
- Provides an ordered view of all updates to an image

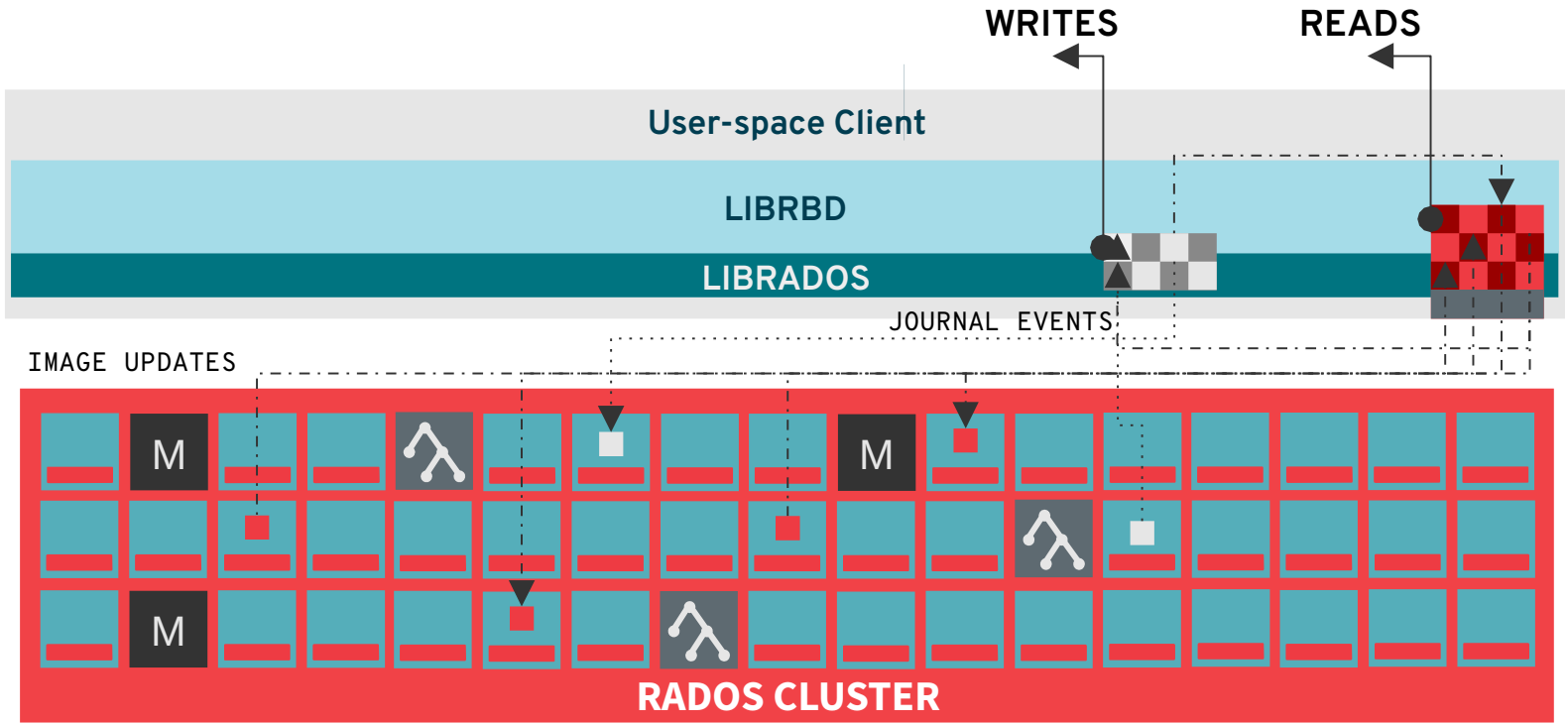
JOURNAL DESIGN

- Typical IO path with journaling:
  a. Create an event to describe the update
  b. Asynchronously append event to journal object
  c. Update in-memory image cache
  d. Blocks cache writeback for affected extent
  e. Completes IO to client
  f. Unblock writeback once event is safe

JOURNAL DESIGN

- IO path with journaling w/o cache:
  a. Create an event to describe the update
  b. Asynchronously append event to journal object
  c. Asynchronously update image once event is safe
  d. Complete IO to client once update is safe

JOURNAL DESIGN



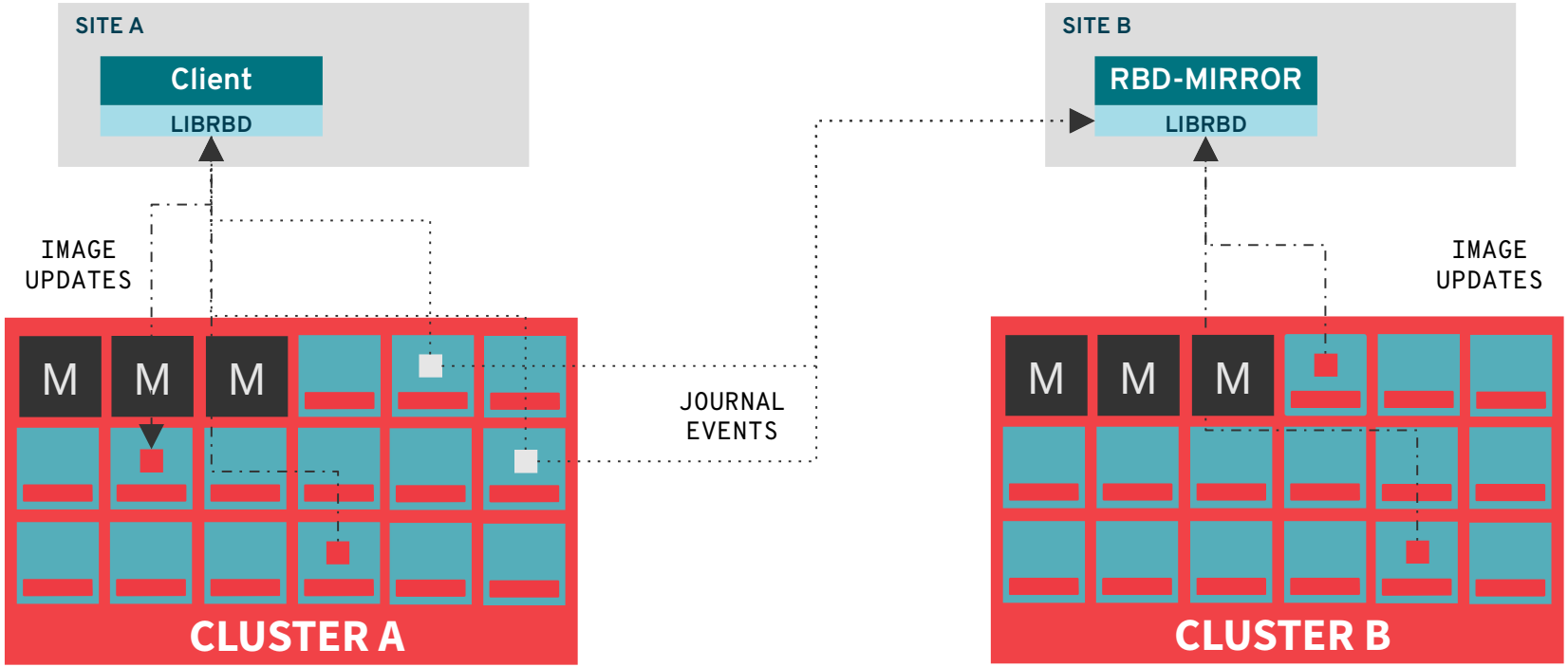
MIRRORING OVERVIEW

- Mirroring peers are configured per-pool
- Enabling mirroring is a per-image property
- Requires that the journal image feature is enabled
- Image is primary (R/W) or non-primary (R/O)
- Replication is handled by rbd-mirror daemon

RBD-MIRROR DAEMON

- Requires access to remote and local clusters
- Responsible for image (re)sync
- Replays journal to achieve consistency
  - Pulls journal event from remote cluster
  - Applies event to local cluster
  - Commits journal event
  - Trim journal
- Transparently handles failover/failback
- Two-way replication between two sites
- One-way replication between N sites

RBD-MIRROR DAEMON



MIRRORING SETUP

- Deploy rbd-mirror daemon on each cluster
  - Jewel/Kraken only support single active daemon per-cluster
- Provide uniquely named “ceph.conf” for each remote cluster
- Create pools with same name on each cluster
- Enable pool mirroring (rbd mirror pool enable)
- Specify peer cluster via rbd CLI (rbd mirror pool peer add)
- Enable image journaling feature (rbd feature enable journaling)
- Enable image mirroring (rbd mirror image enable)

SITE FAILOVER

- Per-image failover / failback
- Coordinated demotion / promotion
  - (rbd mirror image demote)
  - (rbd mirror image promote)
- Uncoordinated promotion + resync
  - (rbd mirror image promote --force)
- Resync from force-promotion / split-brain
  - (rbd mirror image resync)

CAVEATS

- Write IOs have worst-case 2x performance hit
  - Journal event append
  - Image object write
- In-memory cache can mask hit if working set fits
- Only supported by librbd-based clients

MITIGATION

- Use a small SSD/NVMe-backed pool for journals
  - ‘rbd journal pool = <fast pool name>’
- Batch multiple events into a single journal append
  - ‘rbd journal object flush age = <seconds>’
- Increase journal data width to match queue depth
  - ‘rbd journal splay width = <number of objects>’
- Future work: potentially parallelize journal append + image write between write barriers

FUTURE FEATURES

- Active/Active rbd-mirror daemons
- Deferred replication and deletion
- “Deep Scrub” of replicated images
- Smarter image resynchronization
- Improved health status reporting
- Improved pool promotion process

BE PREPARED

- Design for failure
- Ceph now provides additional tools for full-scale disaster recovery
- RBD workloads can seamlessly relocate between geographic sites
- **Feedback is welcome**

Questions?

THANK YOU!

Jason Dillaman
RBD Project Tech Lead

dillaman@redhat.com