

RED HAT
SUMMIT

BOSTON, MA
JUNE 23-26, 2015

Ceph Block Devices: A Deep Dive

Josh Durgin
RBD Lead
June 24, 2015

Ceph Motivating Principles

- All components must scale horizontally
- There can be no single point of failure
- The solution must be hardware agnostic
- Should use commodity hardware
- Self-manage wherever possible
- Open Source (LGPL)
- Move beyond legacy approaches
 - client/cluster instead of client/server
 - Ad hoc HA

Ceph Components

APP



RGW

A web services gateway for object storage, compatible with S3 and Swift

HOST/VM



RBD

A reliable, fully-distributed block device with cloud platform integration

CLIENT



CEPHFS

A distributed file system with POSIX semantics and scale-out metadata management

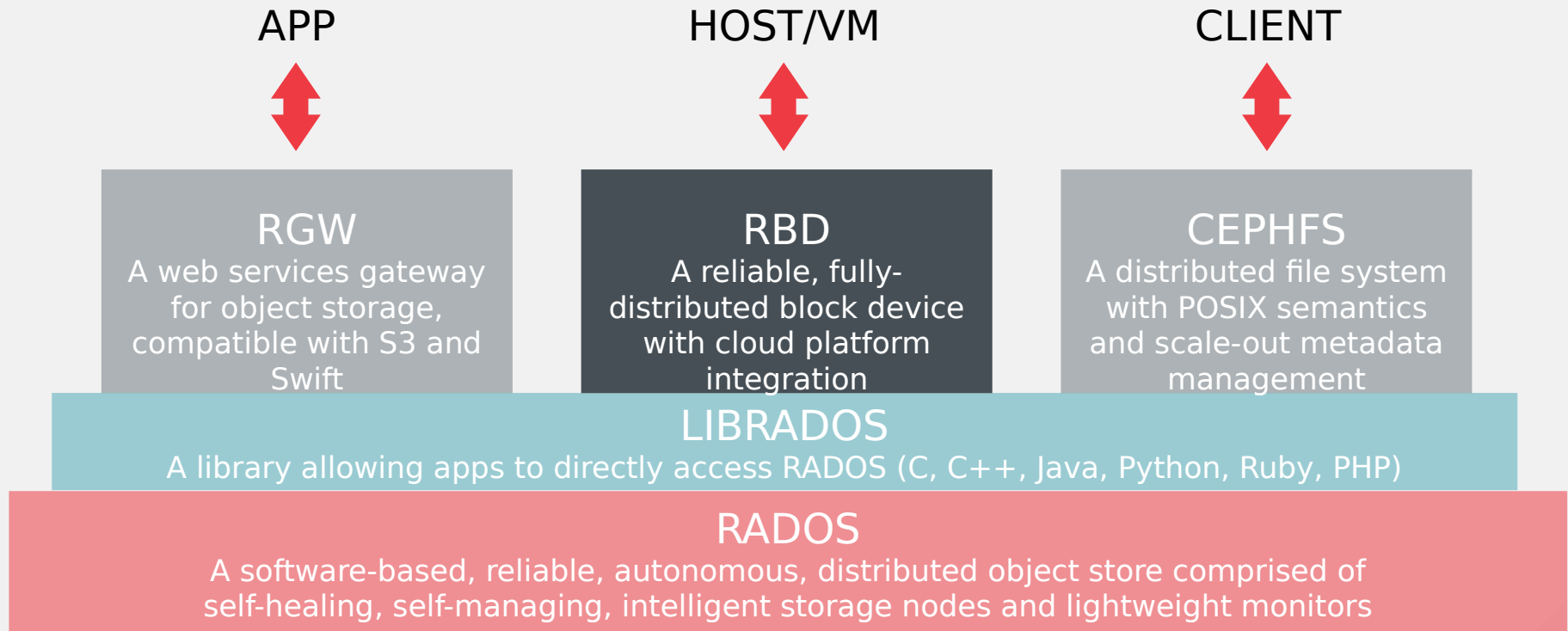
LIBRADOS

A library allowing apps to directly access RADOS (C, C++, Java, Python, Ruby, PHP)

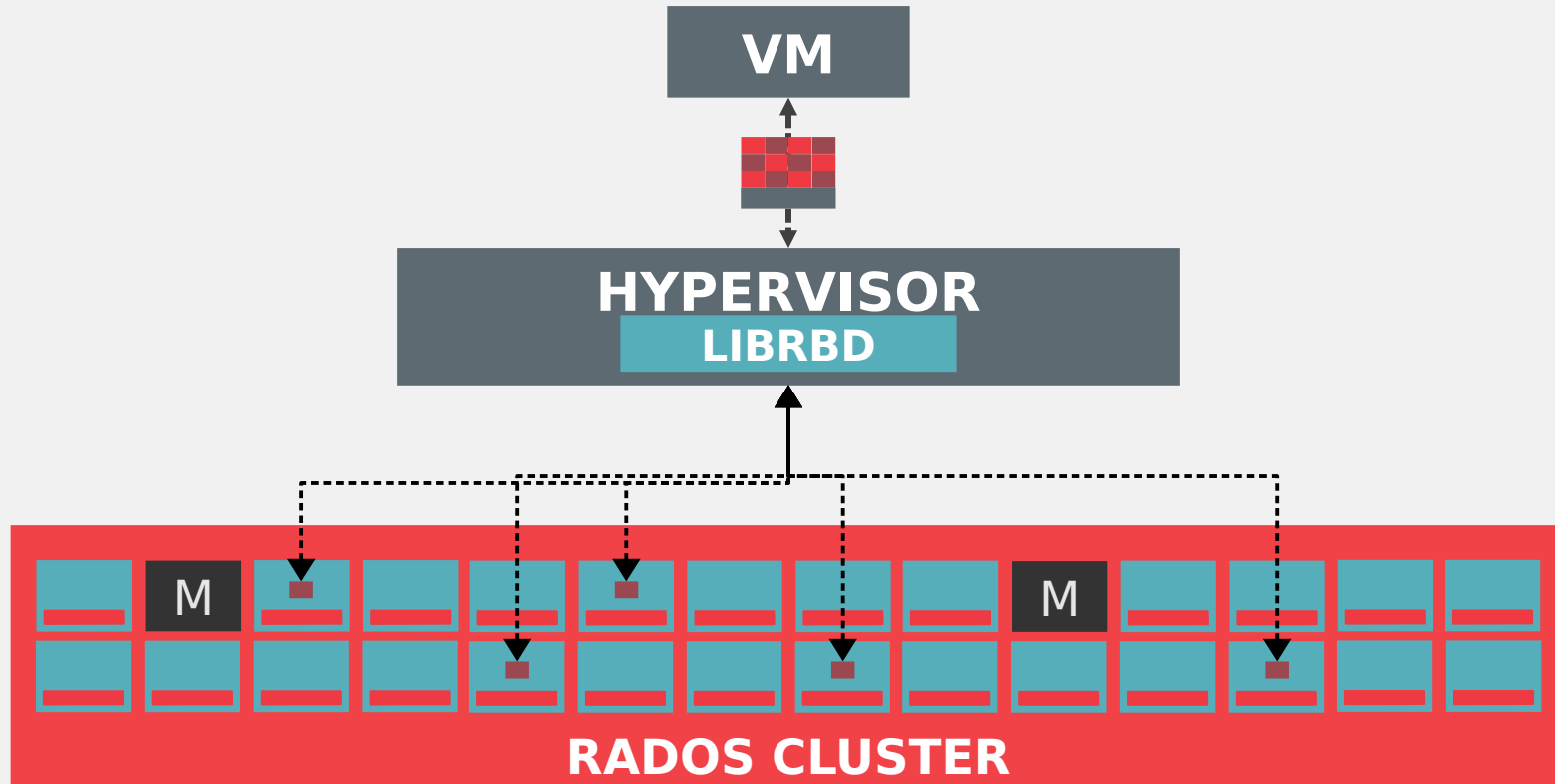
RADOS

A software-based, reliable, autonomous, distributed object store comprised of self-healing, self-managing, intelligent storage nodes and lightweight monitors

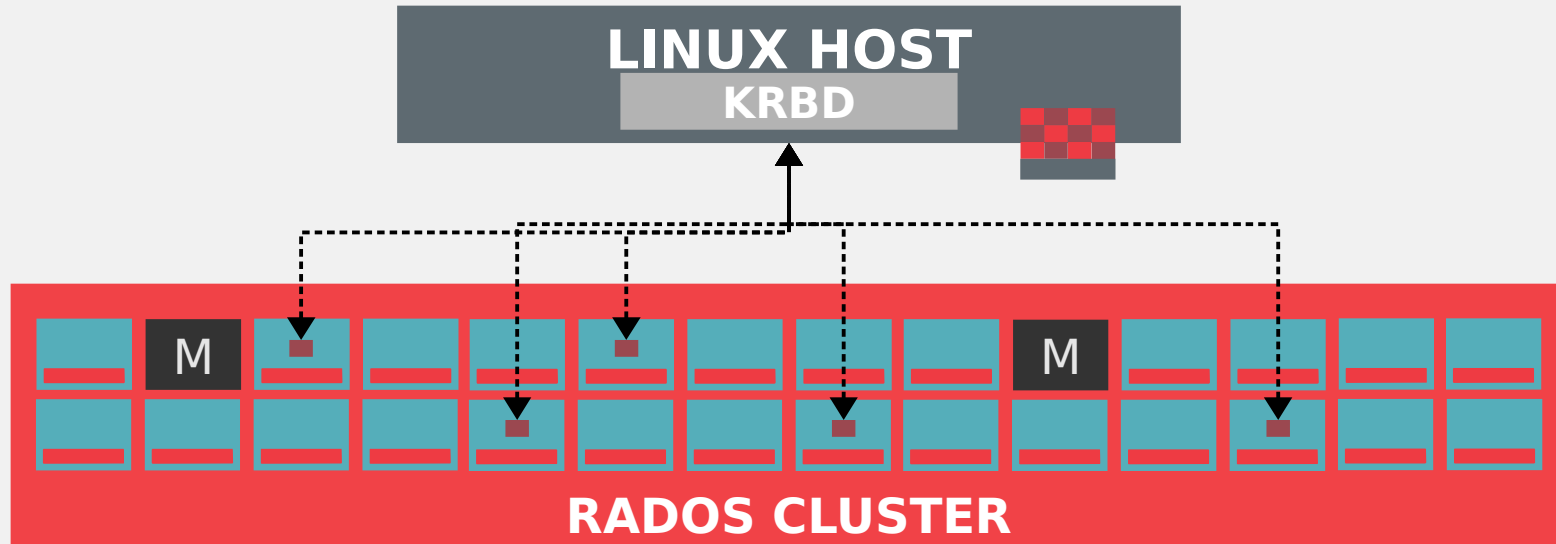
Ceph Components



Storing Virtual Disks

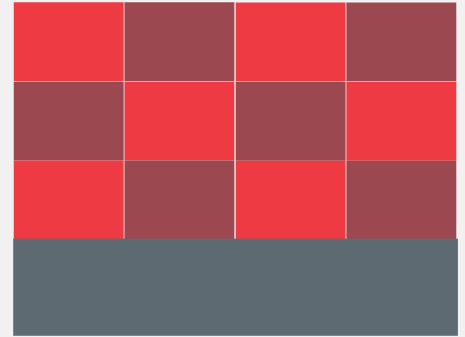


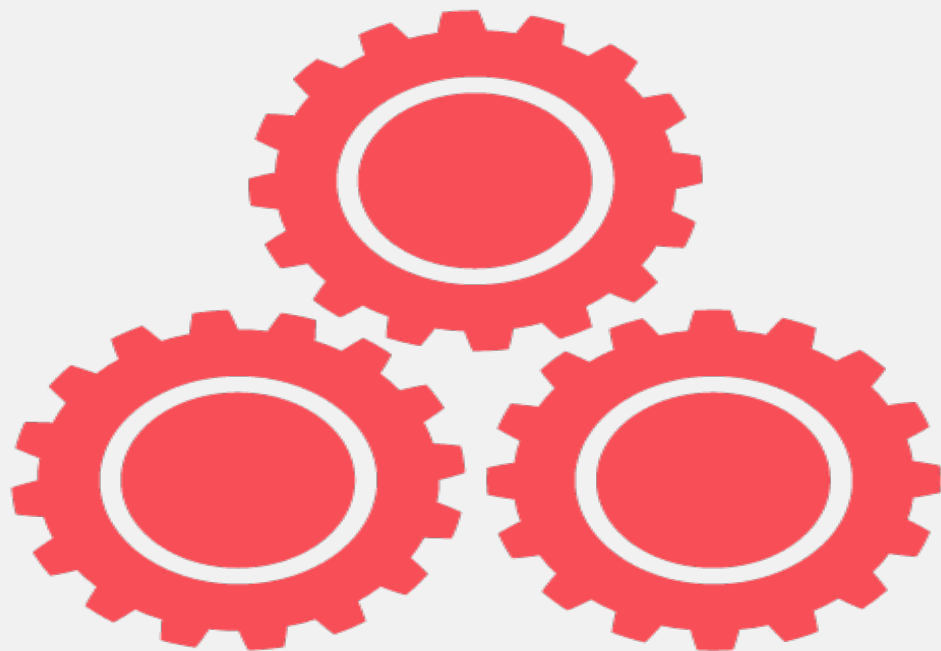
Kernel Module



RBD

- Stripe images across entire cluster (pool)
- Read-only snapshots
- Copy-on-write clones
- Broad integration
 - QEMU, libvirt
 - Linux kernel
 - iSCSI (STGT, LIO)
 - OpenStack, CloudStack, OpenNebula, Ganeti, Proxmox, oVirt
- Incremental backup (relative to snapshots)





Ceph Components

APP



RGW

A web services gateway for object storage, compatible with S3 and Swift

HOST/VM



RBD

A reliable, fully-distributed block device with cloud platform integration

CLIENT



CEPHFS

A distributed file system with POSIX semantics and scale-out metadata management

LIBRADOS

A library allowing apps to directly access RADOS (C, C++, Java, Python, Ruby, PHP)

RADOS

A software-based, reliable, autonomous, distributed object store comprised of self-healing, self-managing, intelligent storage nodes and lightweight monitors

RADOS

- Flat namespace within a pool
- Rich object API
 - Bytes, attributes, key/value data
 - Partial overwrite of existing data
 - Single-object compound atomic operations
 - RADOS classes (stored procedures)
- Strong consistency (CP system)
- Infrastructure aware, dynamic topology
- Hash-based placement (CRUSH)
- Direct client to server data path

RADOS Components



OSDs:

10s to 1000s in a cluster

- One per disk (or one per SSD, RAID group...)
- Serve stored objects to clients
- Intelligently peer for replication & recovery



Monitors:

- Maintain cluster membership and state
- Provide consensus for distributed decision-making
- Small, odd number (e.g., 5)
- Not part of data path

Ceph Components

APP



RGW

A web services gateway for object storage, compatible with S3 and Swift

HOST/VM



RBD

A reliable, fully-distributed block device with cloud platform integration

CLIENT



CEPHFS

A distributed file system with POSIX semantics and scale-out metadata management

LIBRADOS

A library allowing apps to directly access RADOS (C, C++, Java, Python, Ruby, PHP)

RADOS

A software-based, reliable, autonomous, distributed object store comprised of self-healing, self-managing, intelligent storage nodes and lightweight monitors

Metadata

- rbd_directory
 - Maps image name to id, and vice versa
- rbd_children
 - Lists clones in a pool, indexed by parent
- rbd_id.\$image_name
 - The internal id, locatable using only the user-specified image name
- rbd_header.\$image_id
 - Per-image metadata

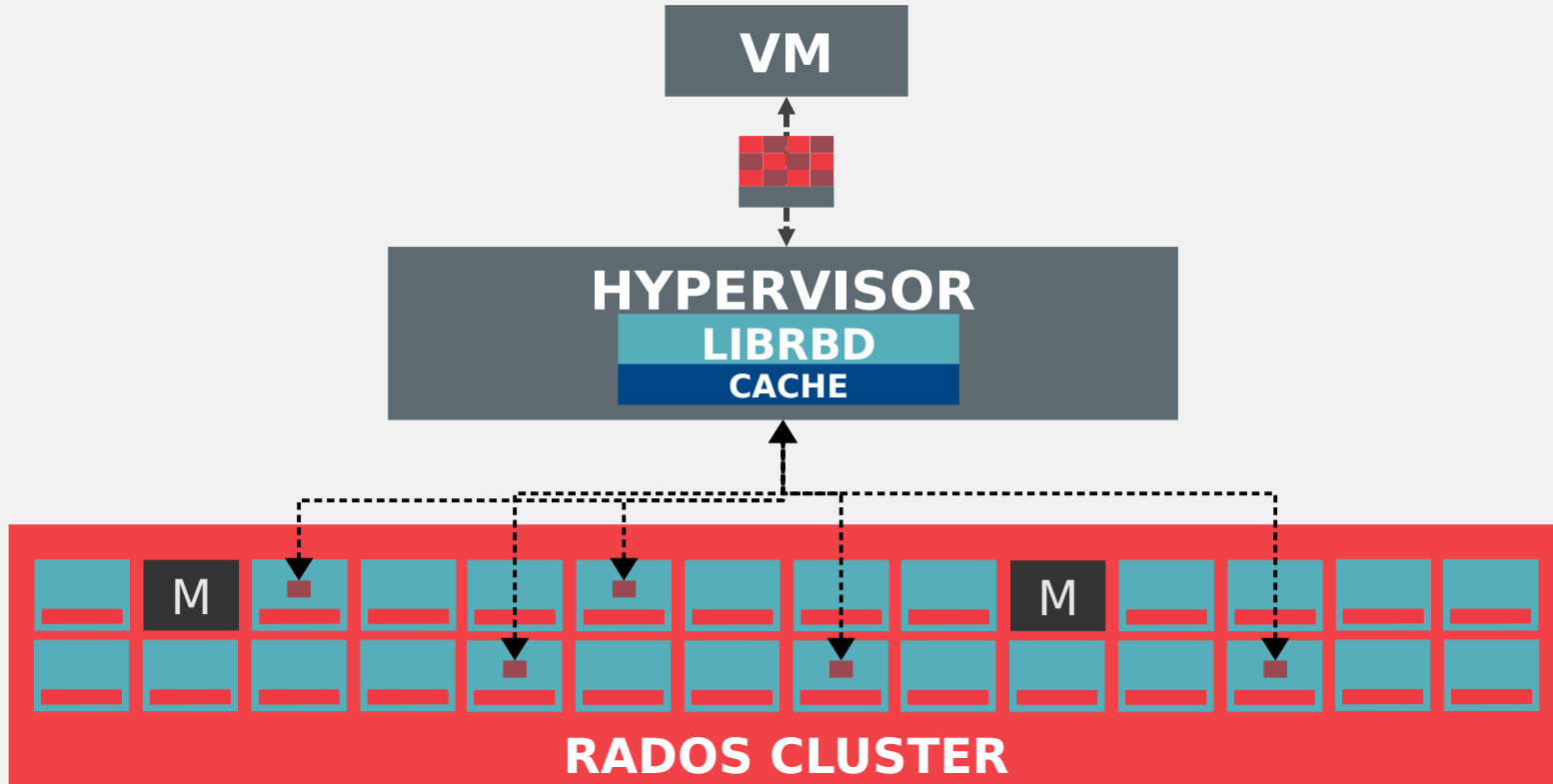
Data

- rbd_data.* objects
 - Named based on offset in image
 - Non-existent to start with
 - Plain data in each object
 - Snapshots handled by rados
 - Often sparse

Striping

- Objects are uniformly sized
 - Default is simple 4MB divisions of device
- Randomly distributed among OSDs by CRUSH
- Parallel work is spread across many spindles
- No single set of servers responsible for image
- Small objects lower OSD usage variance

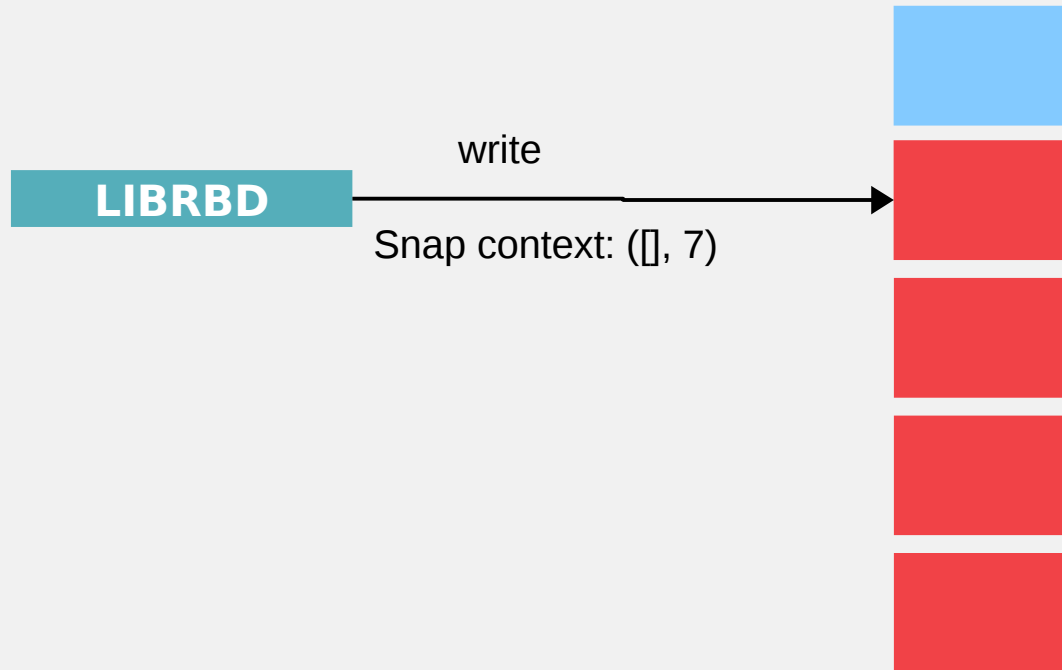
I/O



Snapshots

- Object granularity
- Snapshot context [list of snap ids, latest snap id]
 - Stored in rbd image header (self-managed)
 - Sent with every write
- Snapshot ids managed by monitors
- Deleted asynchronously
- RADOS keeps per-object overwrite stats, so diffs are easy

Snapshots



Snapshots

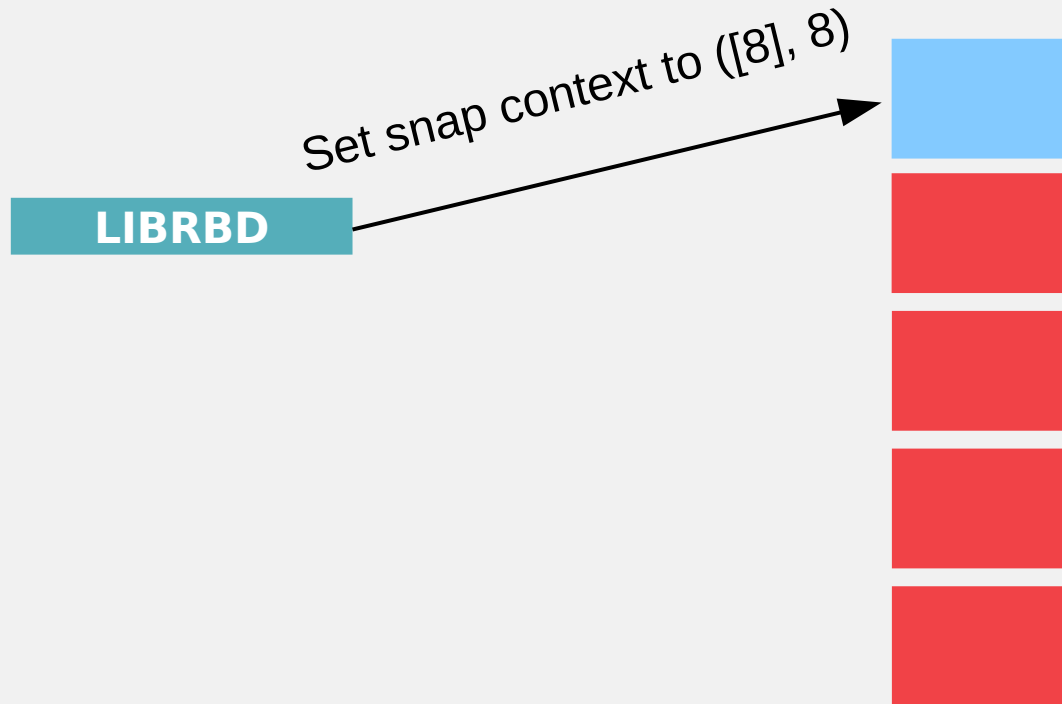
Create snap 8



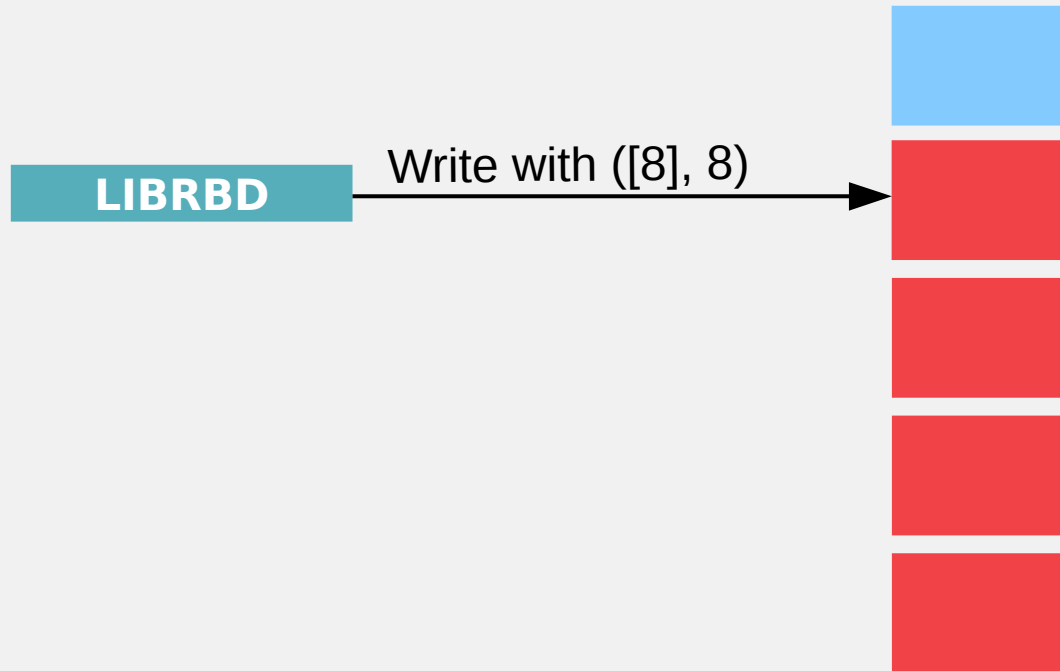
LIBRBD



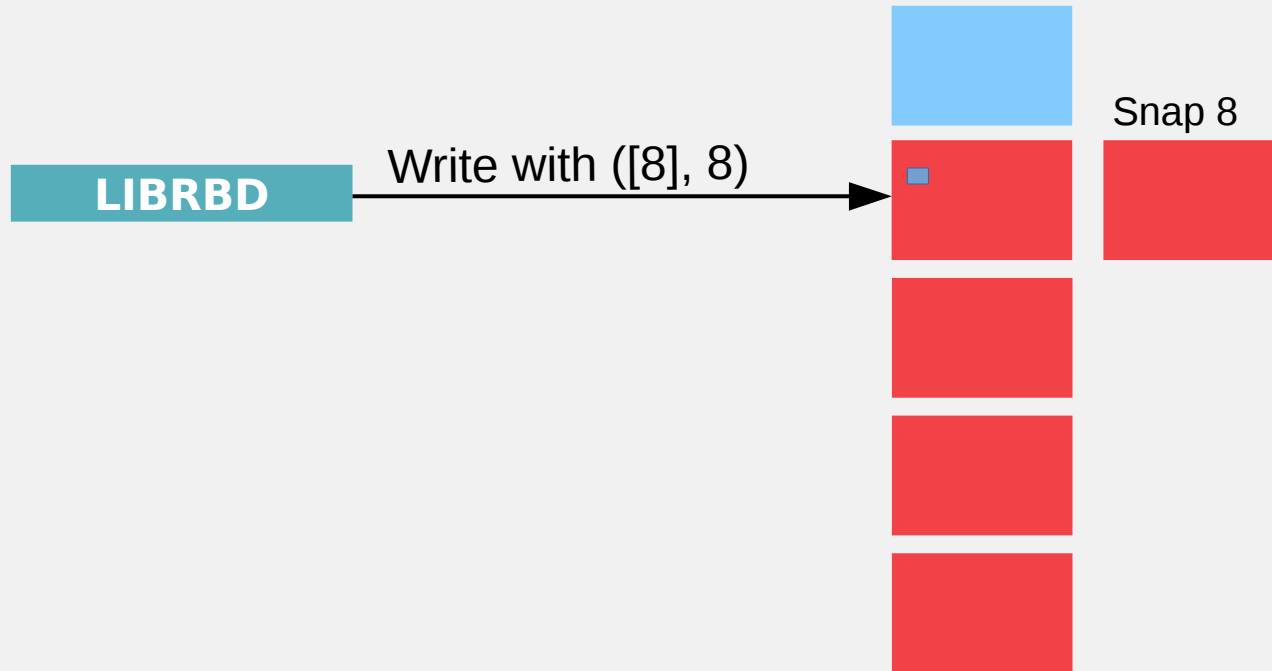
Snapshots



Snapshots



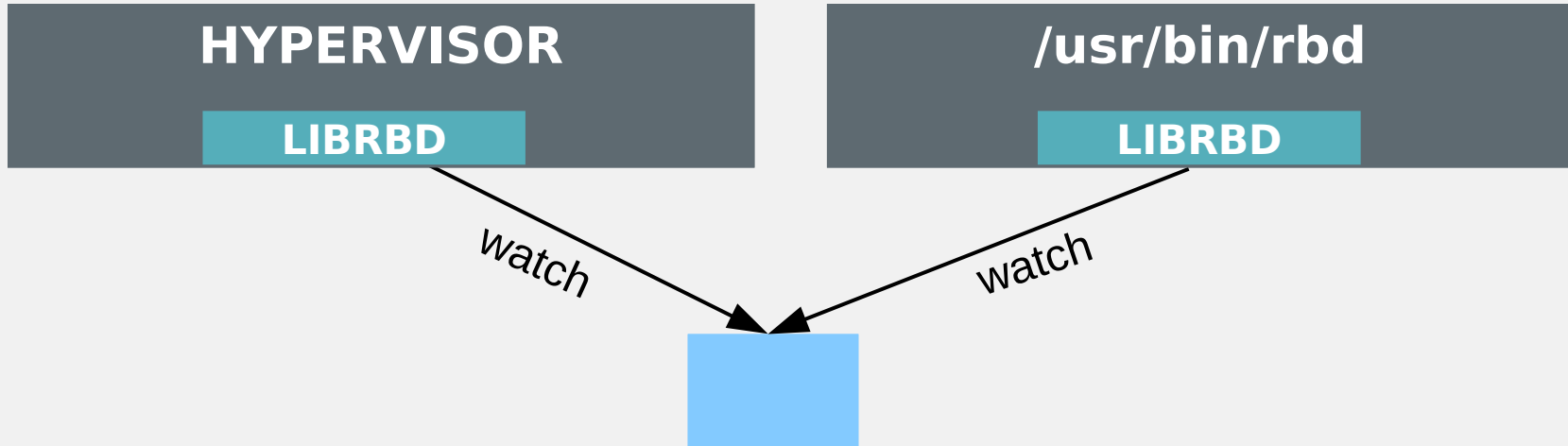
Snapshots



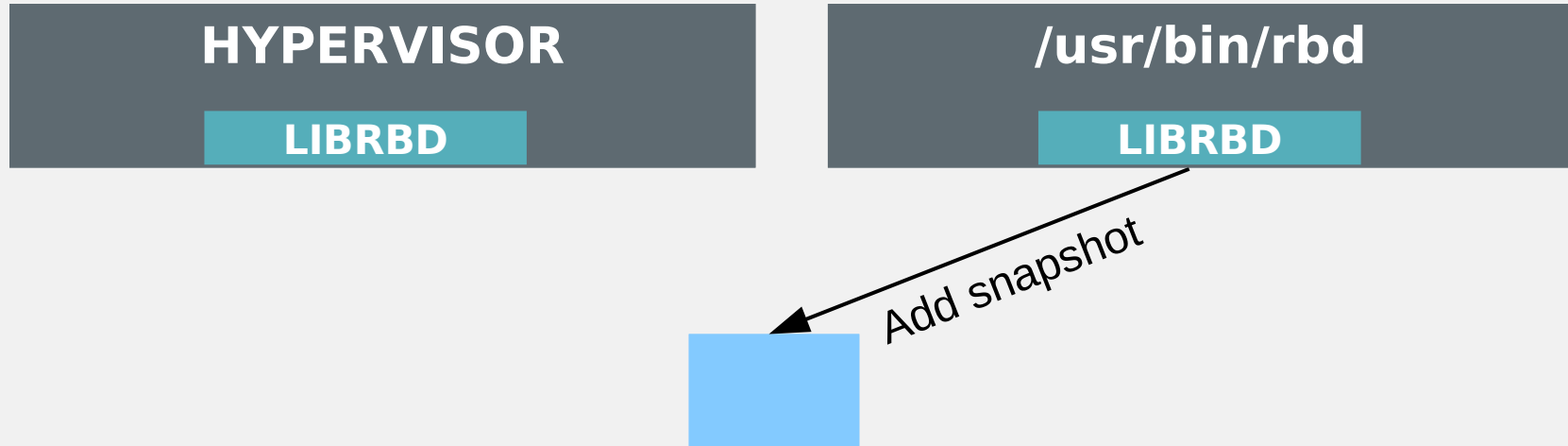
Watch/Notify

- establish stateful 'watch' on an object
 - client interest persistently registered with object
 - client keeps connection to OSD open
- send 'notify' messages to all watchers
 - notify message (and payload) sent to all watchers
 - notification (and reply payloads) on completion
- strictly time-bounded liveness check on watch
 - no notifier falsely believes we got a message

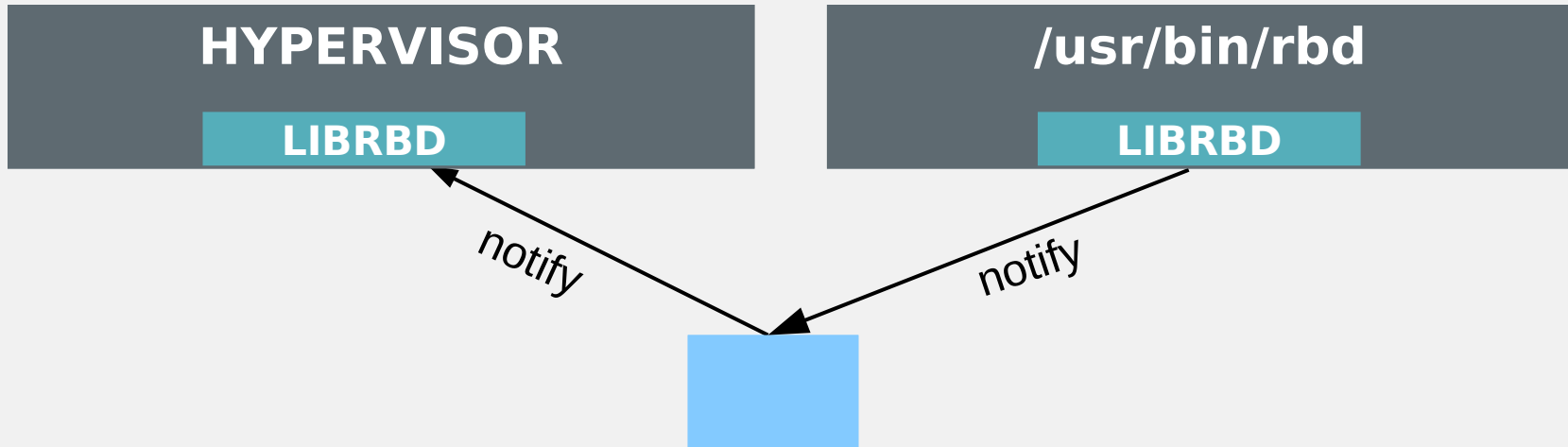
Watch/Notify



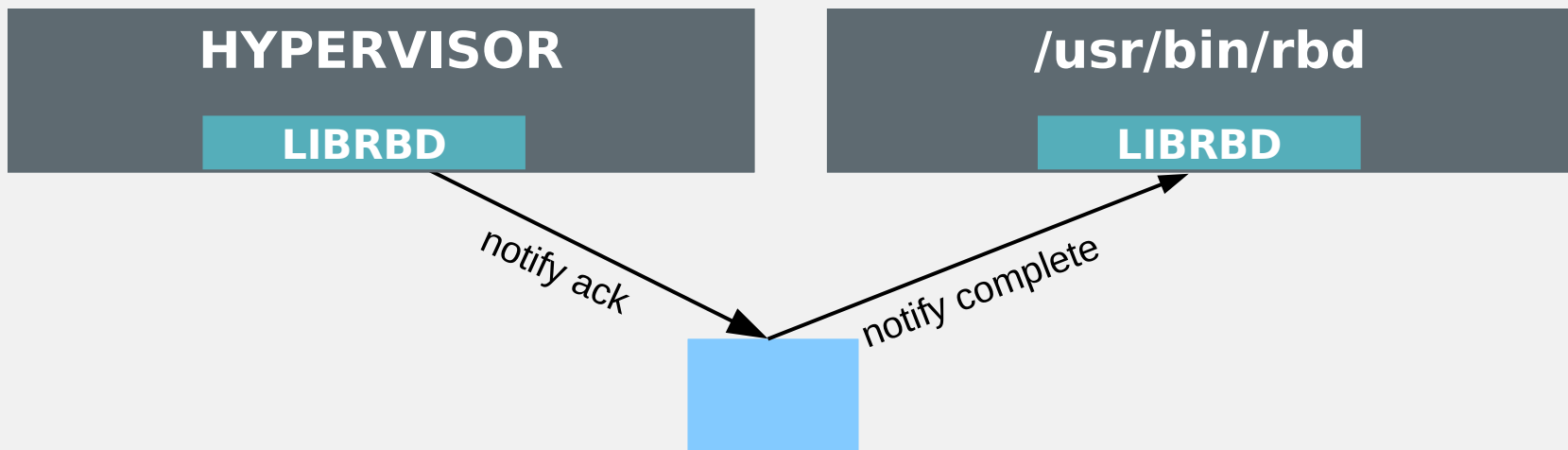
Watch/Notify



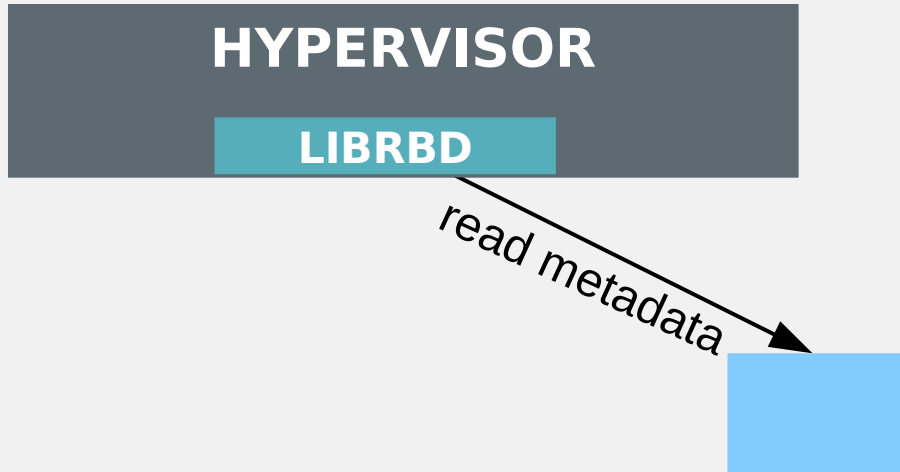
Watch/Notify



Watch/Notify



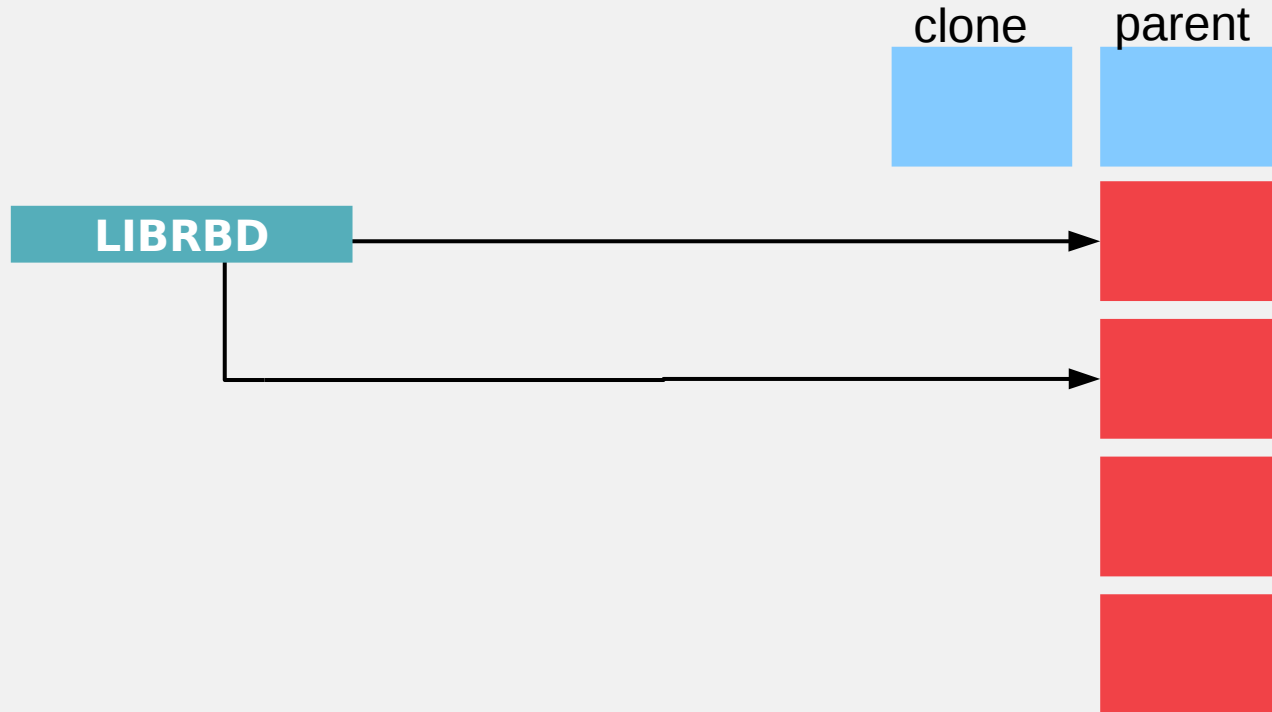
Watch/Notify



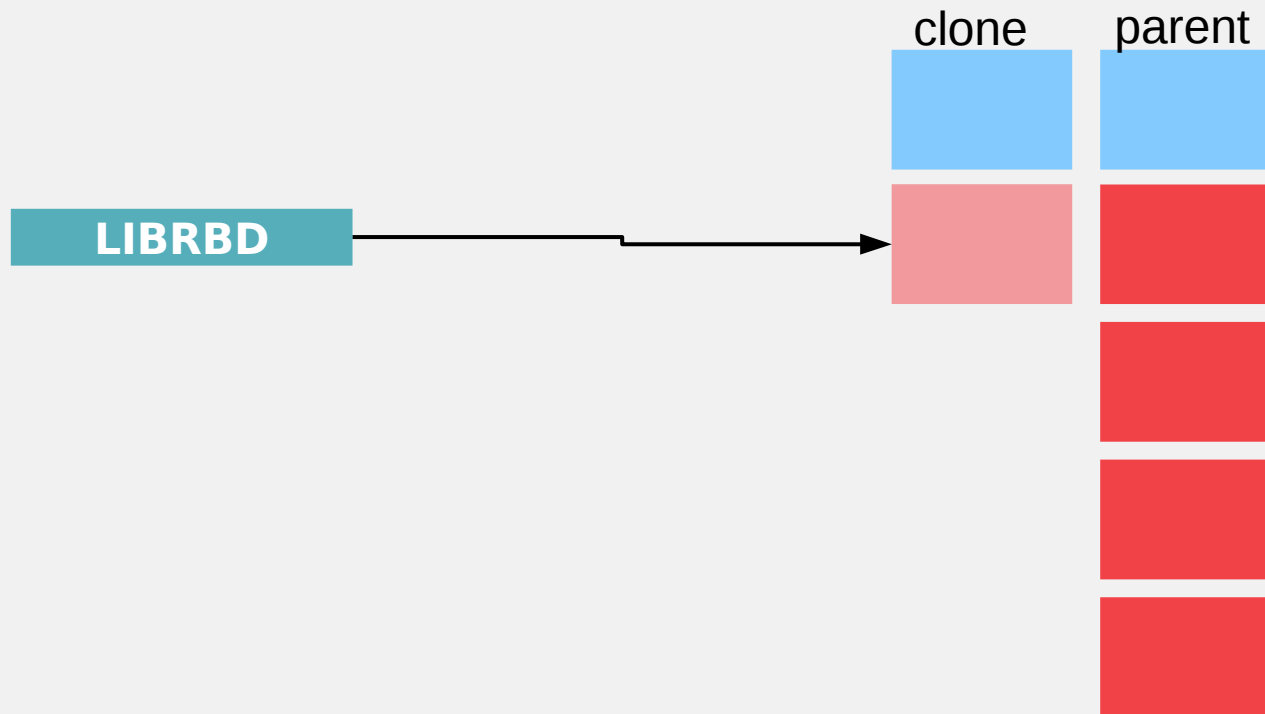
Clones

- Copy-on-write (and optionally read, in Hammer)
- Object granularity
- Independent settings
 - striping, feature bits, object size can differ
 - can be in different pools
- Clones are based on protected snapshots
 - 'protected' means they can't be deleted
- Can be flattened
 - Copy all data from parent
 - Remove parent relationship

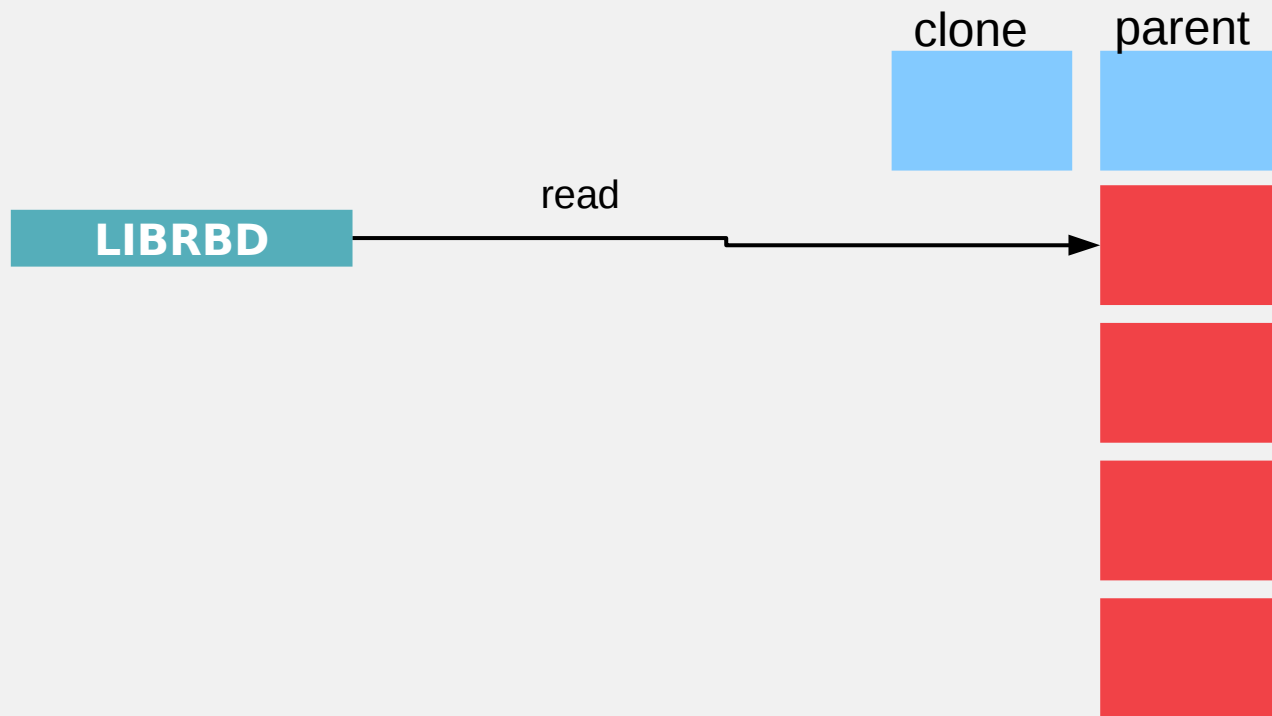
Clones - read



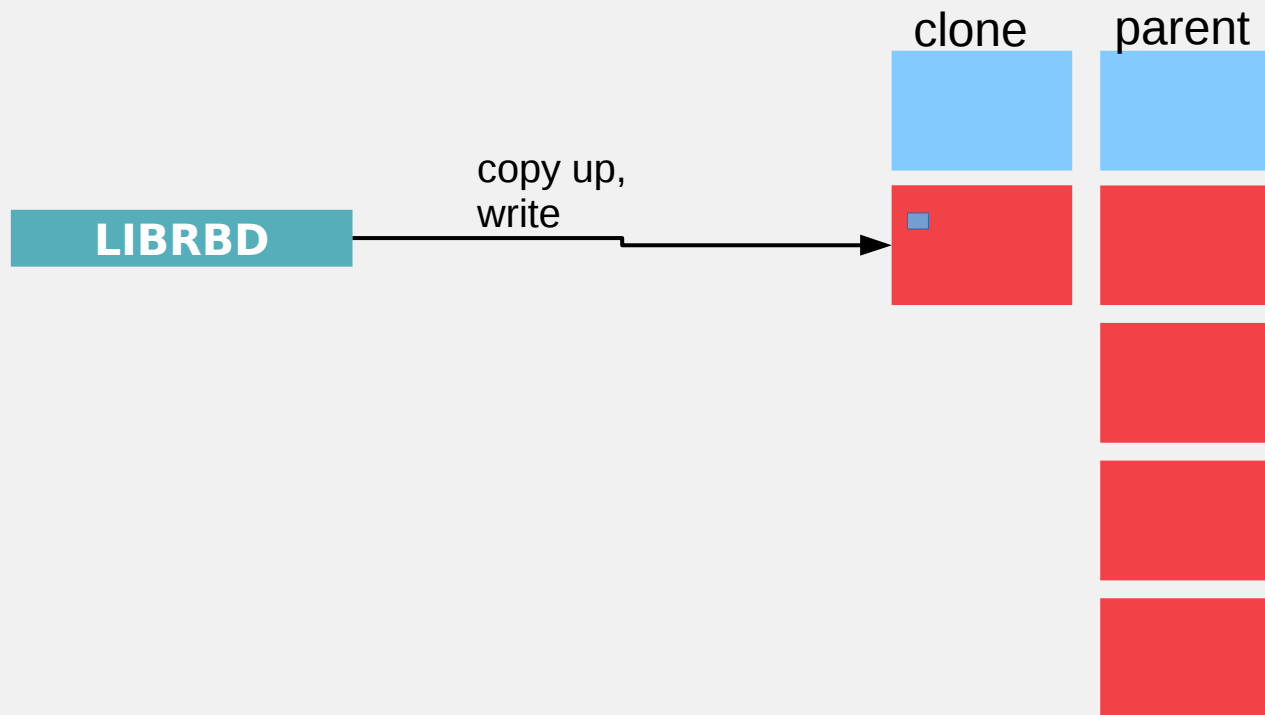
Clones - write



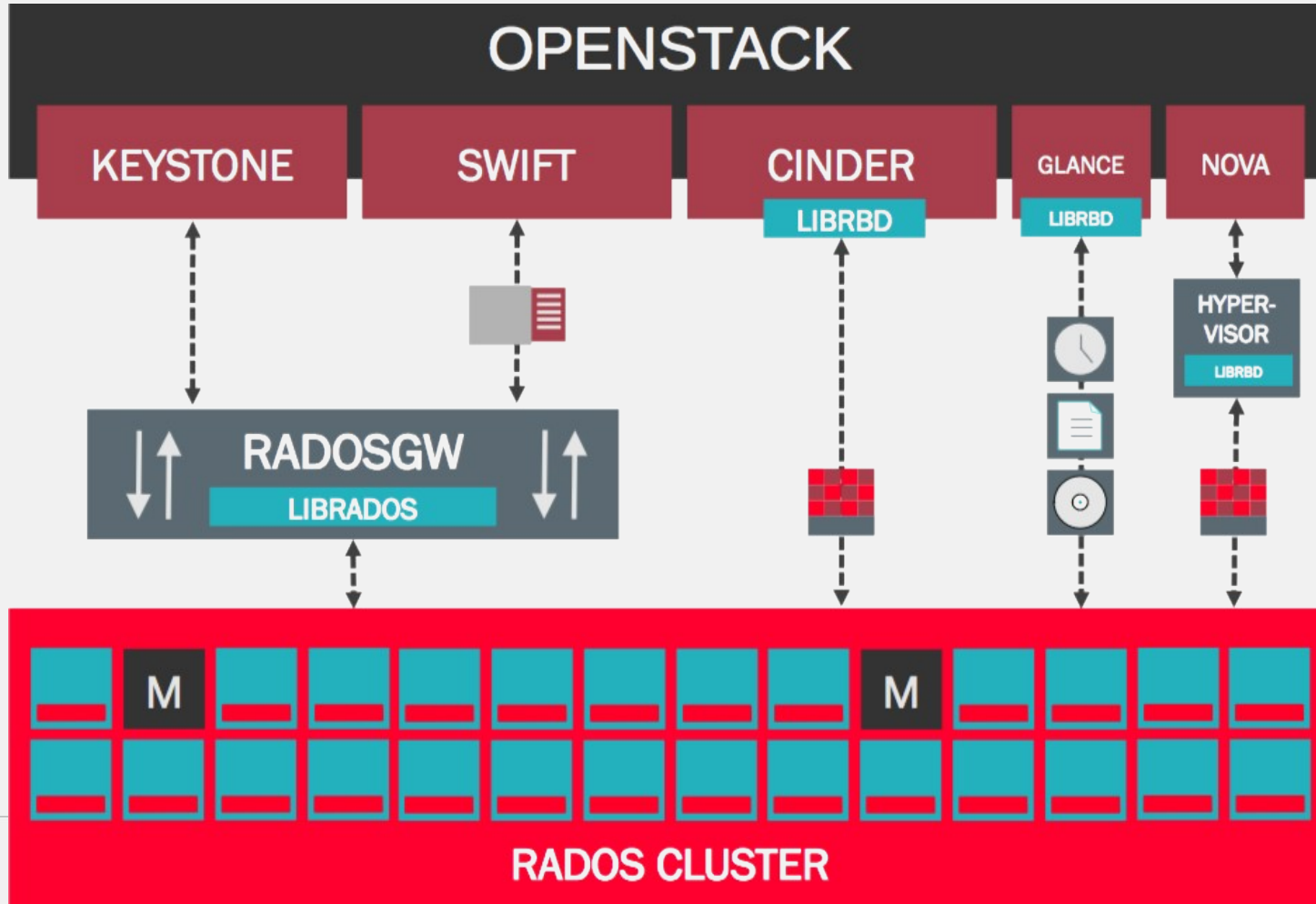
Clones - write



Clones - write



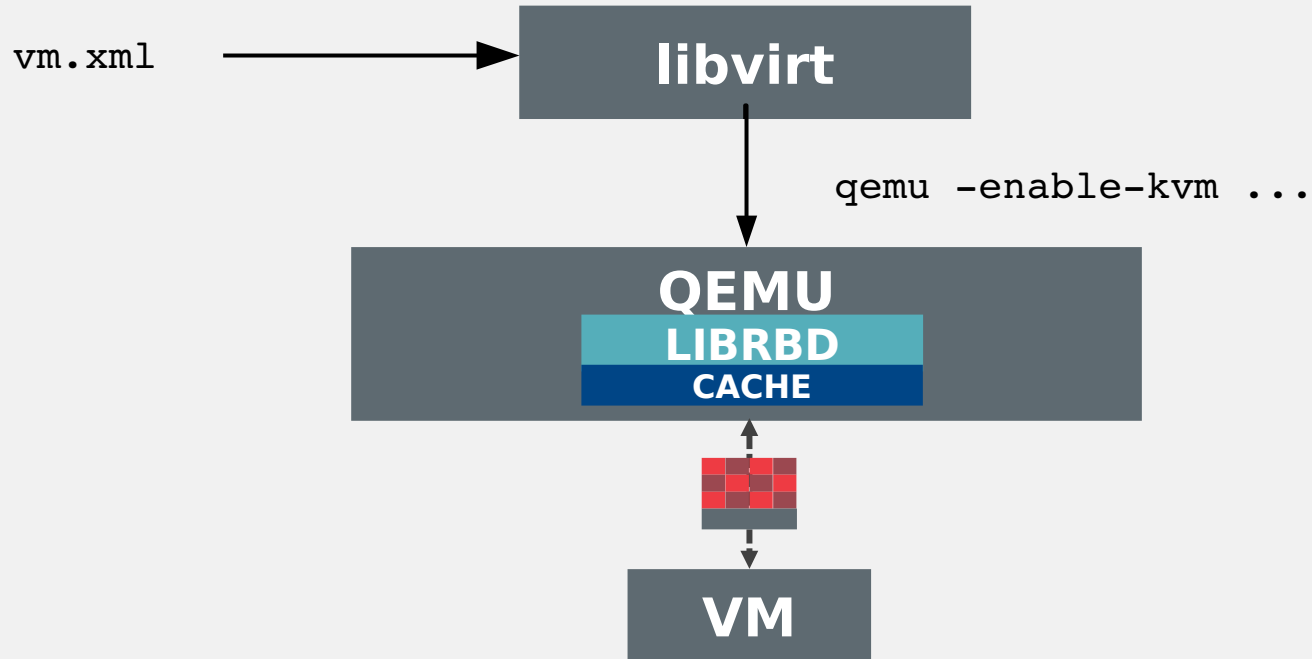
Ceph and OpenStack



Virtualized Setup

- Secret key stored by libvirt
- XML defining VM fed in, includes
 - Monitor addresses
 - Client name
 - QEMU block device cache setting
 - Writeback recommended
 - Bus on which to attach block device
 - virtio-blk/virtio-scsi recommended
 - Ide ok for legacy systems
 - Discard options (ide/scsi only), I/O throttling if desired

Virtualized Setup



Kernel RBD

- `rbd map` sets everything up
- `/etc/ceph/rbdmap` is like `/etc/fstab`
- `udev` adds handy symlinks:
 - `/dev/rbd/$pool/$image[@snap]`
- striping v2 and later feature bits not supported yet
- Can be used to back LIO, NFS, SMB, etc.
- No specialized cache, page cache used by filesystem on top



What's new in Hammer

- Copy-on-read
- Librbd cache enabled in safe mode by default
- Readahead during boot
- Lttng tracepoints
- Allocation hints
- Cache hints
- Exclusive locking (off by default for now)
- Object map (off by default for now)

Infernalis

- Easier space tracking (rbd du)
- Faster differential backups
- Per-image metadata
 - Can persist rbd options
- RBD journaling
- Enabling new features on the fly

Future Work

- Kernel client catch up
- RBD mirroring
- Consistency groups
- QoS for rados (policy at rbd level)
- Active/Active iSCSI
- Performance improvements
 - Newstore osd backend
 - Improved cache tiering
 - Finer-grained caching
 - Many more

Questions?

Josh Durgin

 jdurgin@redhat.com

