virtual machine block storage with the ceph distributed storage system

sage weil

xensummit – august 28, 2012

outline

- why you should care
- what is it, what it does
- how it works, how you can use it
  - architecture
  - objects, recovery
- rados block device
  - integration
  - path forward
- who we are, why we do this

ceph

why should you care about another storage system?

requirements, time, cost

ceph

requirements

- diverse storage needs
  - object storage
  - block devices (for VMs) with snapshots, cloning
  - shared file system with POSIX, coherent caches
  - structured data... files, block devices, or objects?
- scale
  - terabytes, petabytes, exabytes
  - heterogeneous hardware
  - reliability and fault tolerance

ceph

time

- ease of administration
- no manual data migration, load balancing
- painless scaling
  - expansion **and** contraction
  - seamless migration

ceph

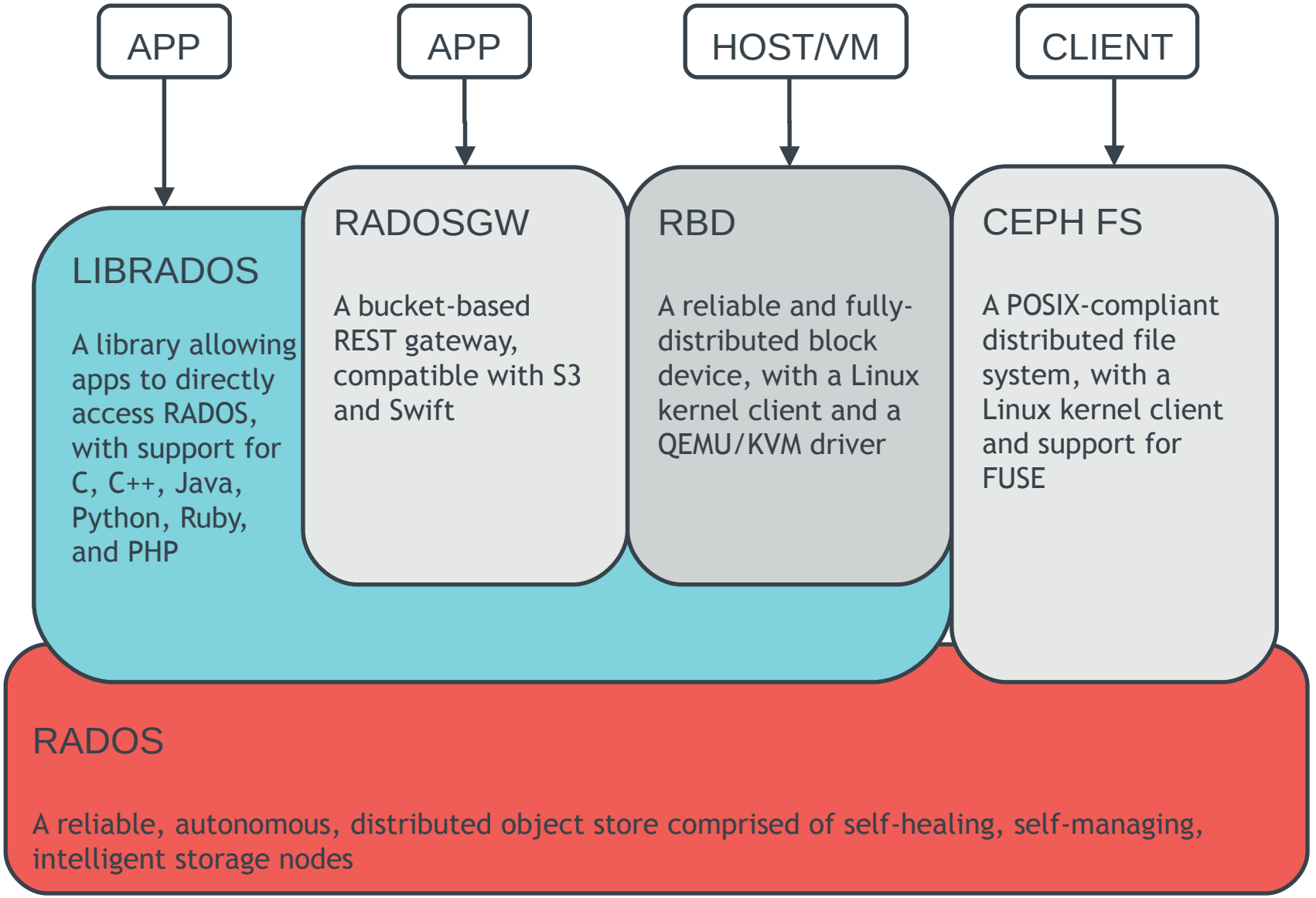
cost

- low cost per gigabyte
- no vendor lock-in

- software solution
  - run on commodity hardware
- open source

ceph

what is ceph?


APP
APP
HOST/VM
CLIENT
LIBRADOS
A library allowing apps to directly access RADOS, with support for C, C++, Java, Python, Ruby, and PHP
RADOSGW
A bucket-based REST gateway, compatible with S3 and Swift
RBD
A reliable and fully-distributed block device, with a Linux kernel client and a QEMU/KVM driver
CEPH FS
A POSIX-compliant distributed file system, with a Linux kernel client and support for FUSE
RADOS
A reliable, autonomous, distributed object store comprised of self-healing, self-managing, intelligent storage nodes

ceph

open source

- LGPLv2
  - copyleft
  - free to link to proprietary code
- no copyright assignment
  - no dual licensing
  - no “enterprise-only” feature set
- active community
- commercial support available

ceph

distributed storage system

- data center (not geo) scale
  - 10s to 10,000s of machines
  - terabytes to exabytes
- fault tolerant
  - no SPoF
  - commodity hardware
    - ethernet, SATA/SAS, HDD/SSD
    - RAID, SAN probably a waste of time, power, and money

ceph

object storage model

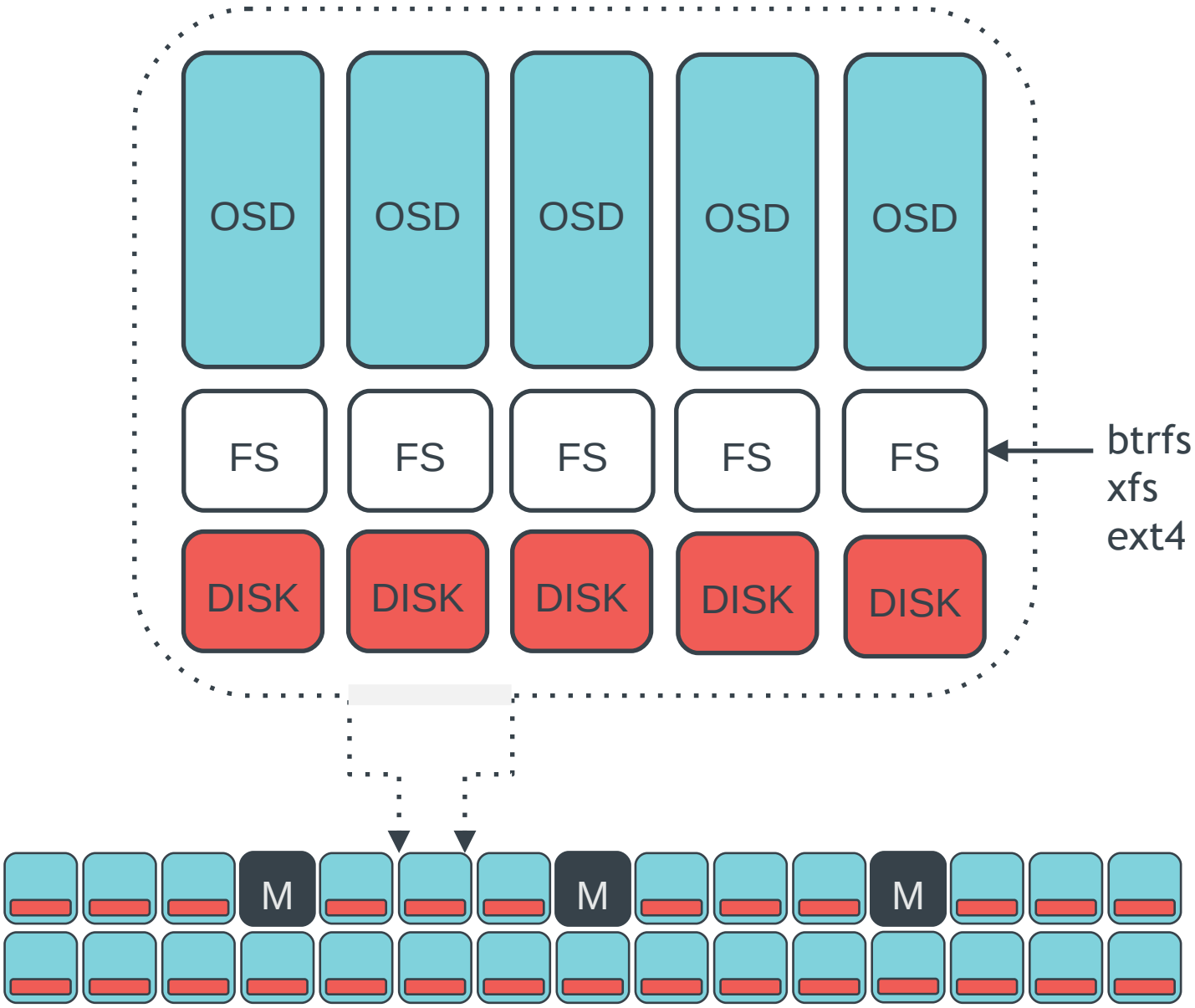
- pools
  - 1s to 100s
  - independent namespaces or object collections
  - replication level, placement policy
- objects
  - trillions
  - blob of data (bytes to gigabytes)
  - attributes (e.g., “version=12”; bytes to kilobytes)
  - key/value bundle (bytes to gigabytes)

ceph

object storage cluster

- conventional client/server model doesn't scale
  - server(s) become bottlenecks; proxies are inefficient
  - if storage devices don't coordinate, clients must
- ceph-osds are intelligent storage daemons
  - coordinate with peers
  - sensible, cluster-aware protocols
  - sit on local file system
    - btrfs, xfs, ext4, etc.
    - leveldb

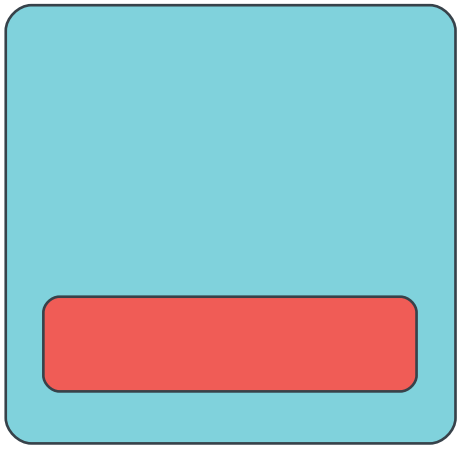
ceph


OSD
OSD
OSD
OSD
OSD
FS
FS
FS
FS
FS
DISK
DISK
DISK
DISK
DISK
btrfs
xfs
ext4
M
M
M

ceph

Monitors:

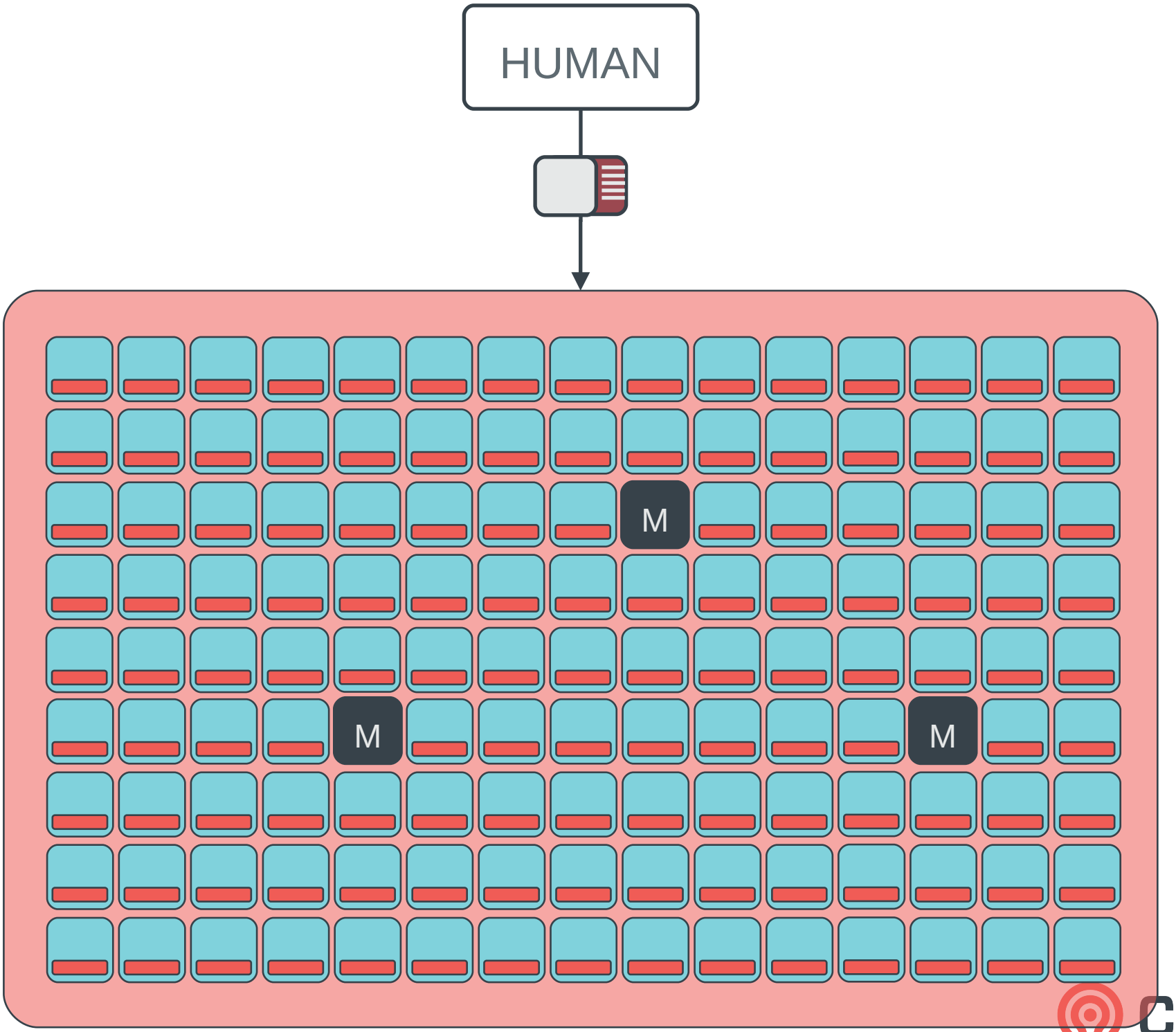
- Maintain cluster state
- Provide consensus for distributed decision-making
- Small, odd number
- These do **not** serve stored objects to clients
-



OSDs:

- One per disk or RAID group
- At least three in a cluster
- Serve stored objects to clients
- Intelligently peer to perform replication tasks

ceph

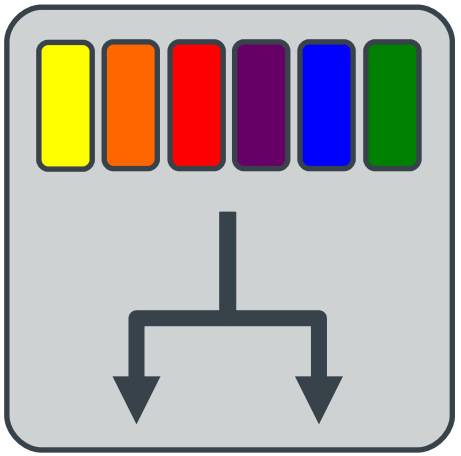


ceph

data distribution

- all objects are replicated N times
- objects are automatically placed, balanced, migrated in a dynamic cluster
- must consider physical infrastructure
  - ceph-osds on hosts in racks in rows in data centers
- three approaches
  - pick a spot; remember where you put it
  - pick a spot; write down where you put it
  - calculate where to put it, where to find it

ceph



CRUSH

- Pseudo-random placement algorithm
- Fast calculation, **no lookup**
- Ensures even distribution
- Repeatable, deterministic
- Rule-based configuration
  - specifiable replication
  - infrastructure topology aware
  - allows weighting
- Stable mapping
  - Limited data migration

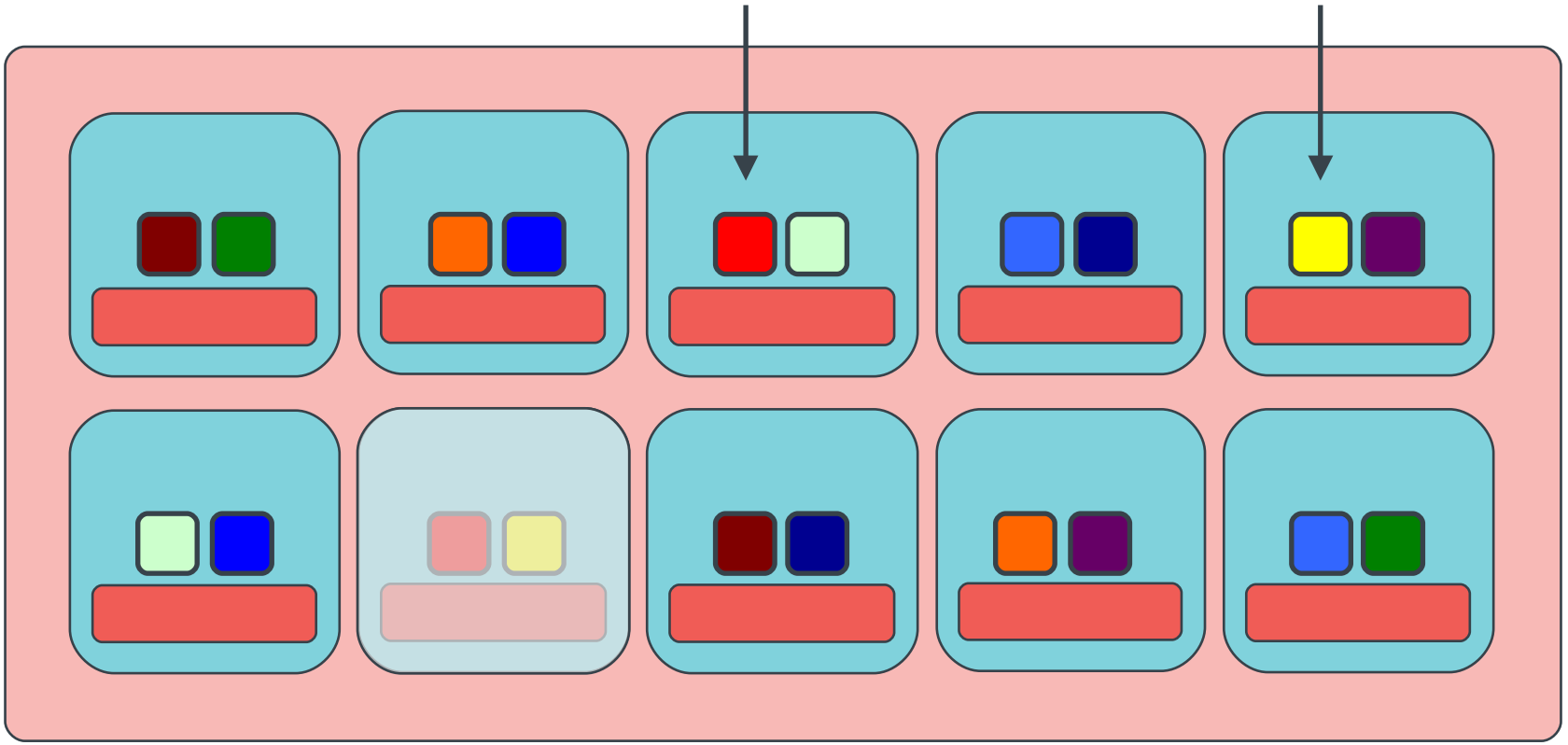
ceph

distributed object storage

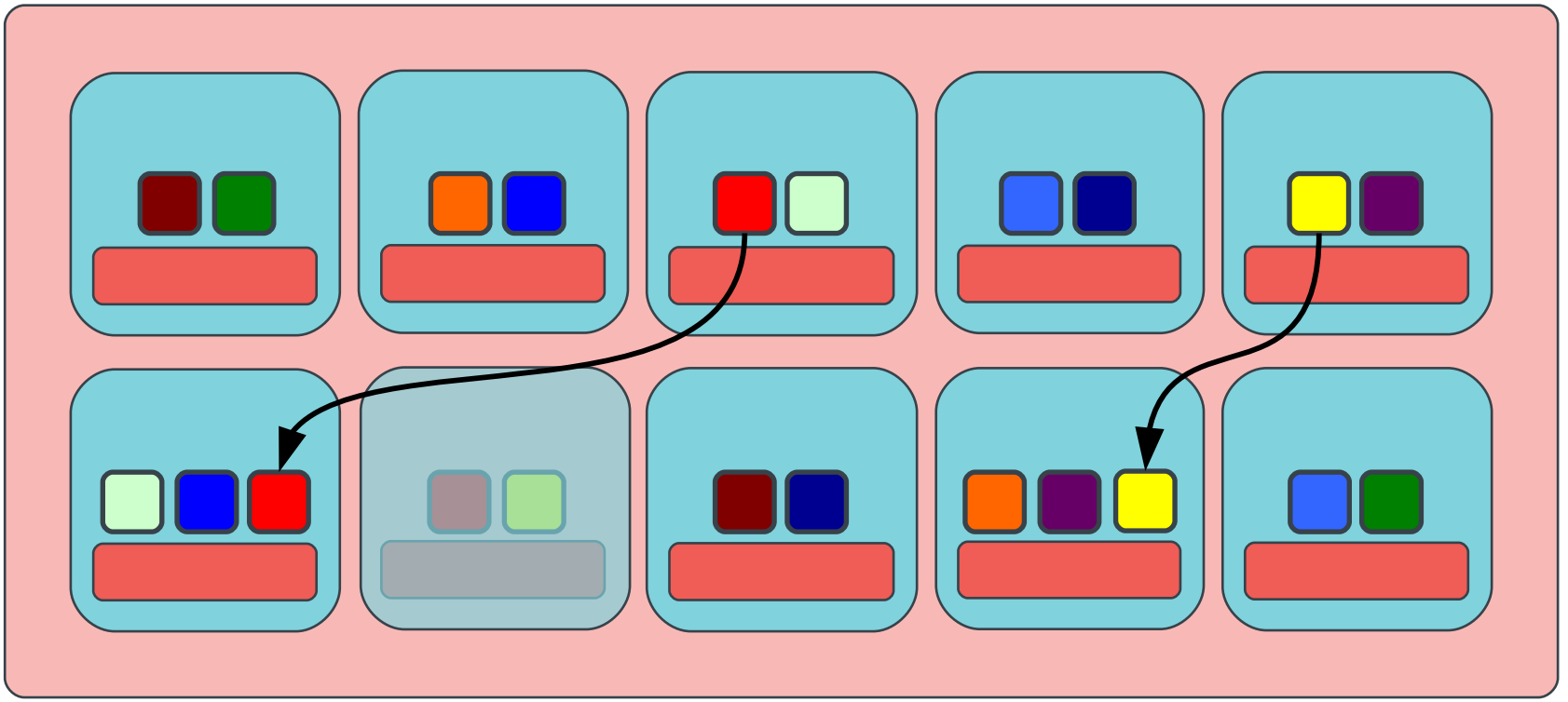
- CRUSH tells us where data should go
  - small “osd map” records cluster state at point in time
  - ceph-osd node status (up/down, weight, IP)
  - CRUSH function specifying desired data distribution
- object storage daemons (RADOS)
  - store it there
  - migrate it as the cluster changes
- decentralized, distributed approach allows
  - massive scales (10,000s of servers or more)
  - efficient data access

large clusters aren't static

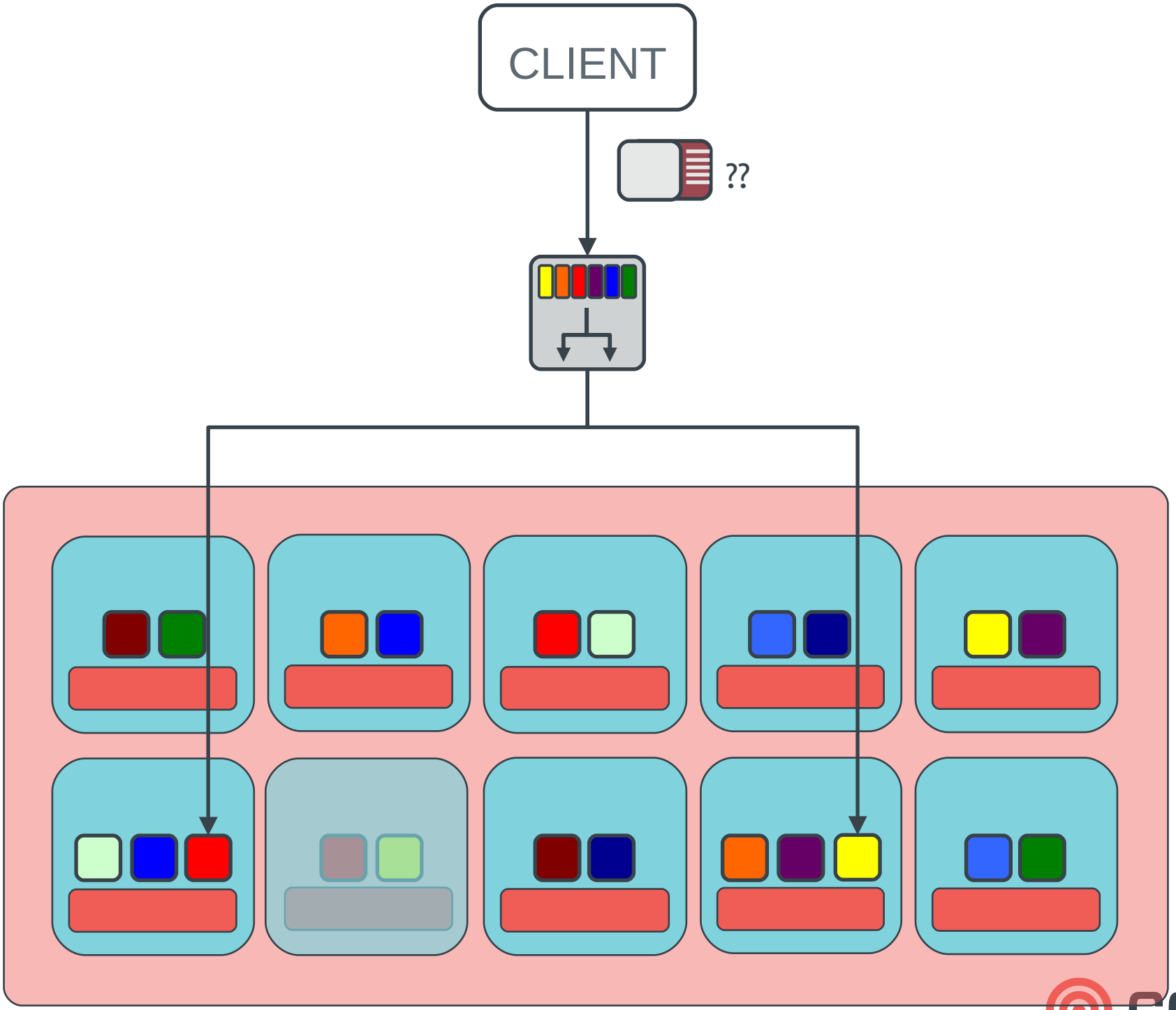
- dynamic cluster
  - nodes are added, removed; nodes reboot, fail, recover
  - recovery is the norm
- osd maps are versioned
  - shared via gossip
- any map update potentially triggers data migration
  - ceph-osds monitor peers for failure
  - new nodes register with monitor
  - administrator adjusts weights, mark out old



ceph



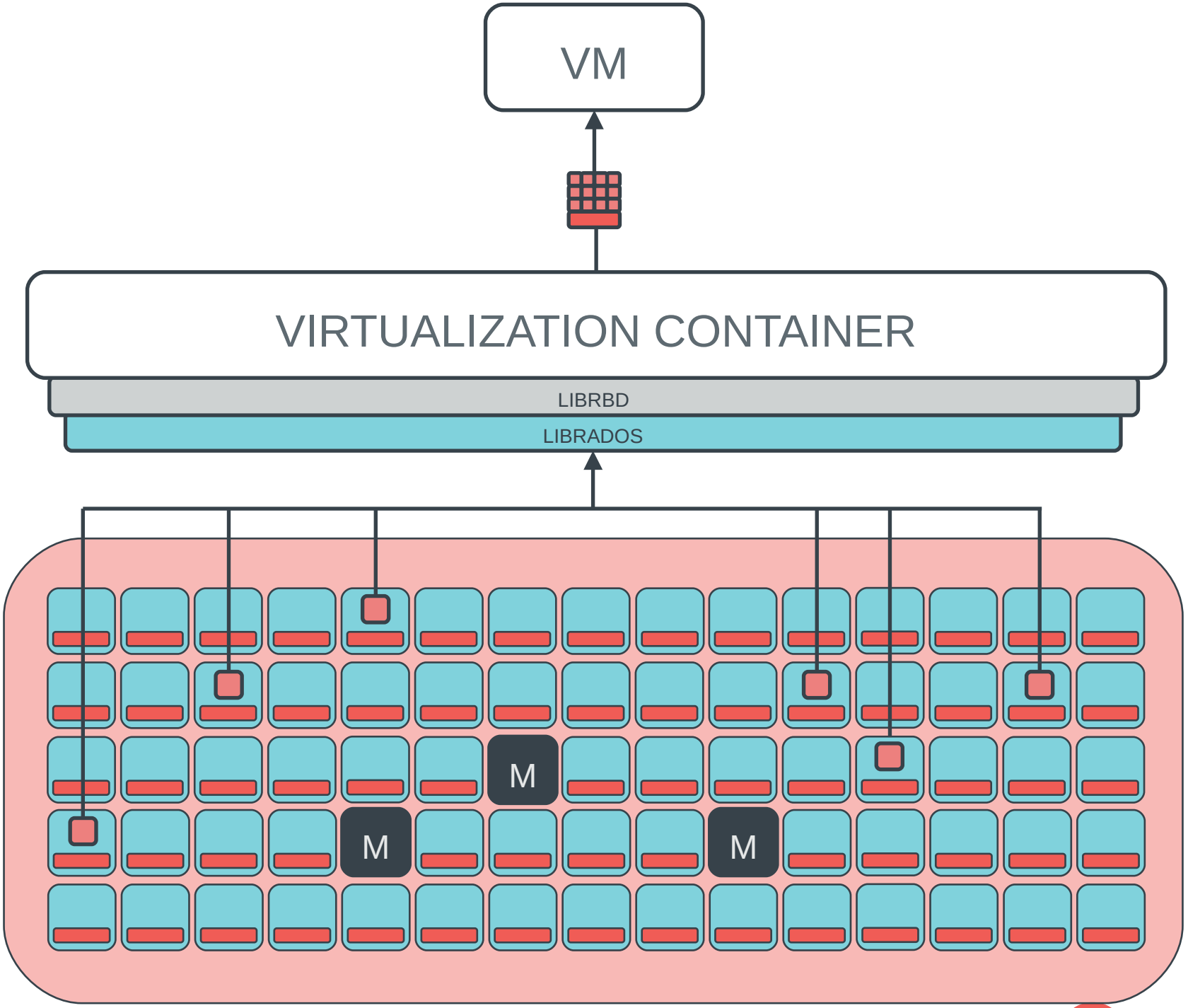
ceph



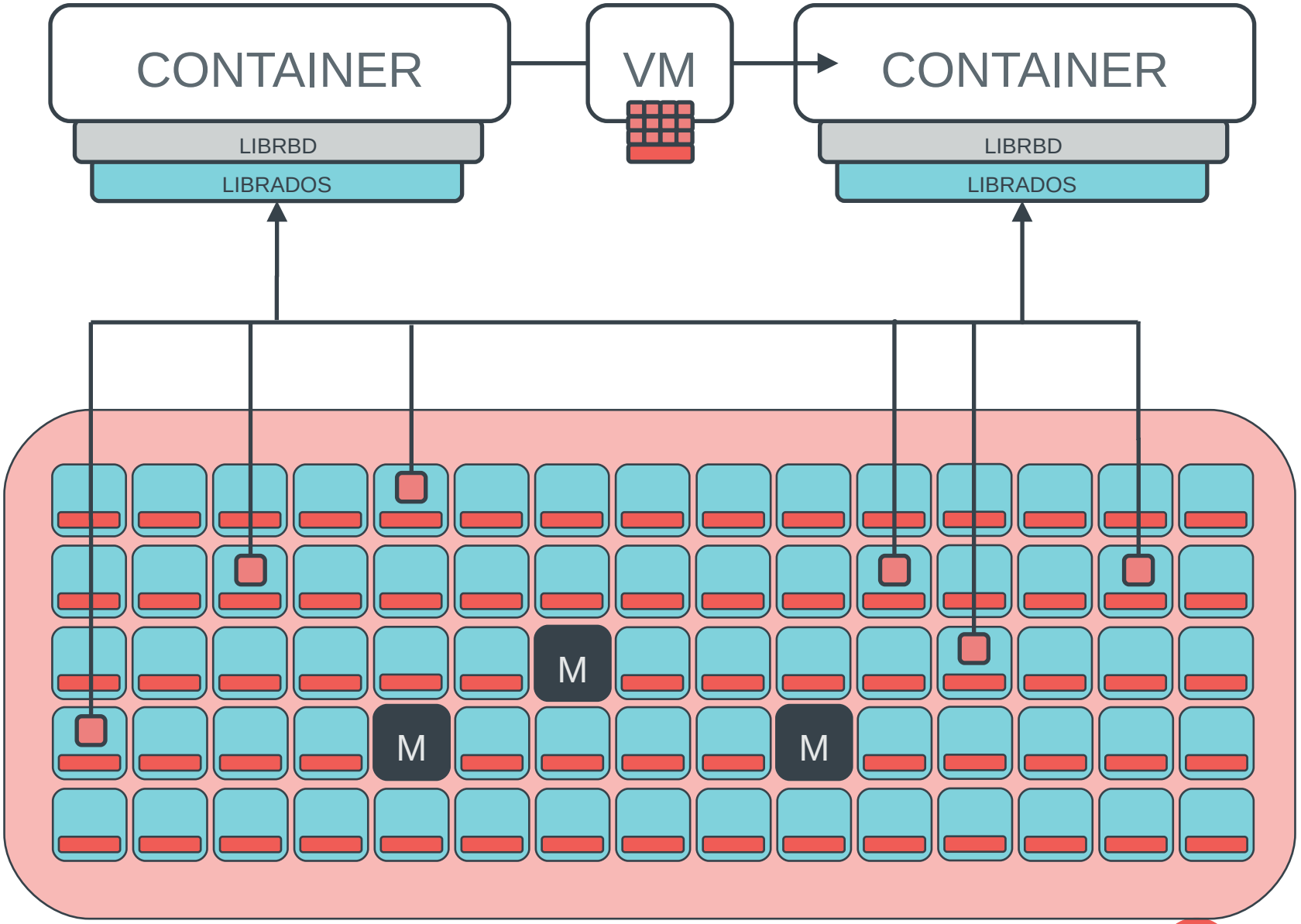
ceph

what does this mean for my cloud?

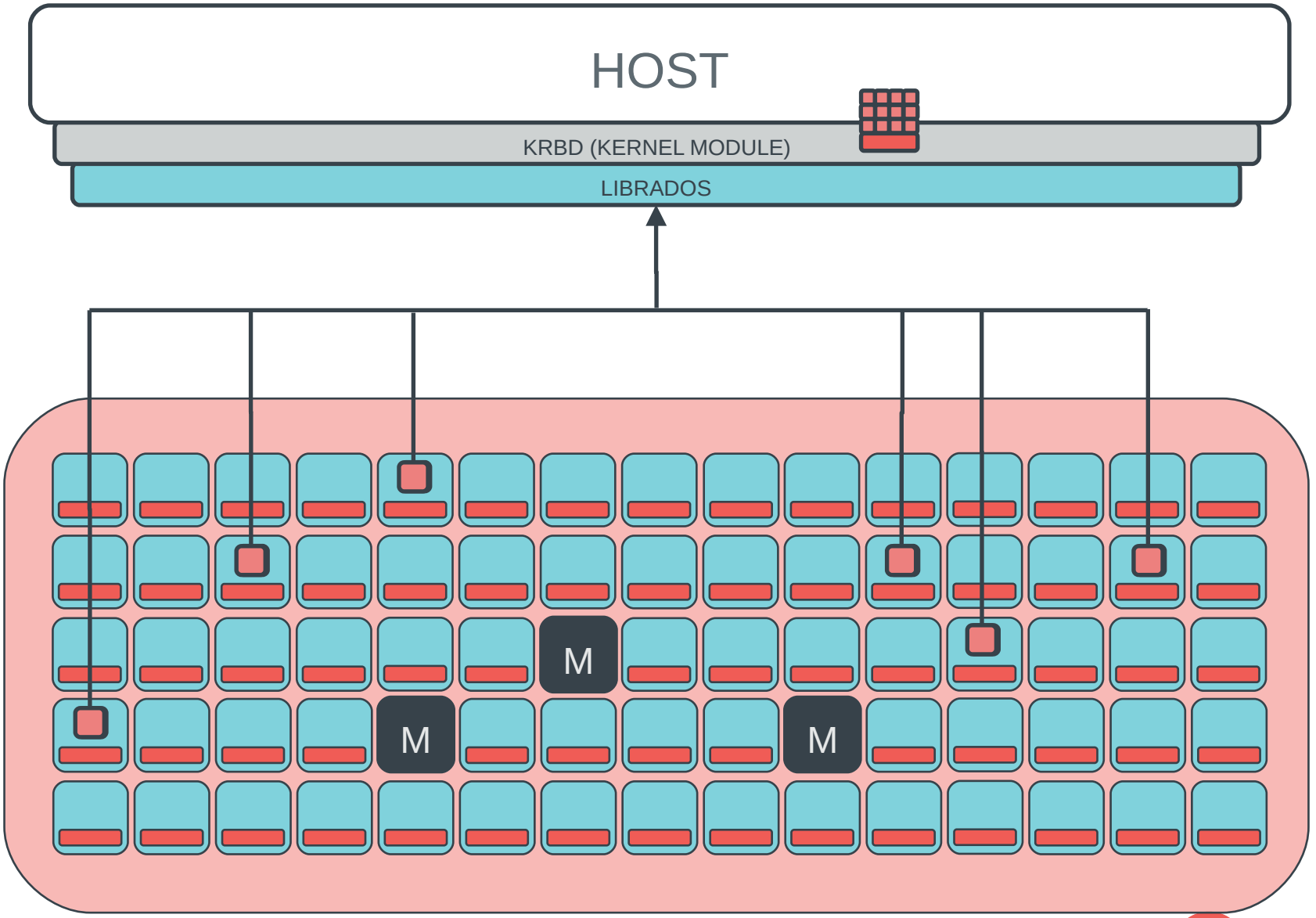
- virtual disks
  - reliable
  - accessible from many hosts
- appliances
  - great for small clouds
  - not viable for public or (large) private clouds
- avoid single server bottlenecks
- efficient management



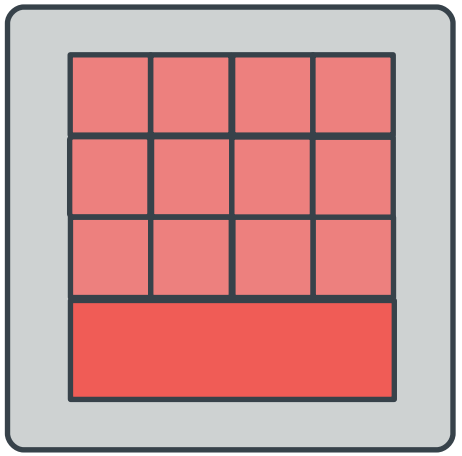
ceph


CONTAINER
VM
CONTAINER
LIBRBD
LIBRADOS
LIBRBD
LIBRADOS
M
M
M

ceph



ceph



RBD: RADOS Block Device

- Replicated, reliable, high-performance virtual disk
- Allows decoupling of VMs and containers
  - Live migration!
- Images are striped across the cluster
- Snapshots!
- Native support in the Linux kernel
  - /dev/rbd1
- librbd allows easy integration

ceph

HOW DO YOU

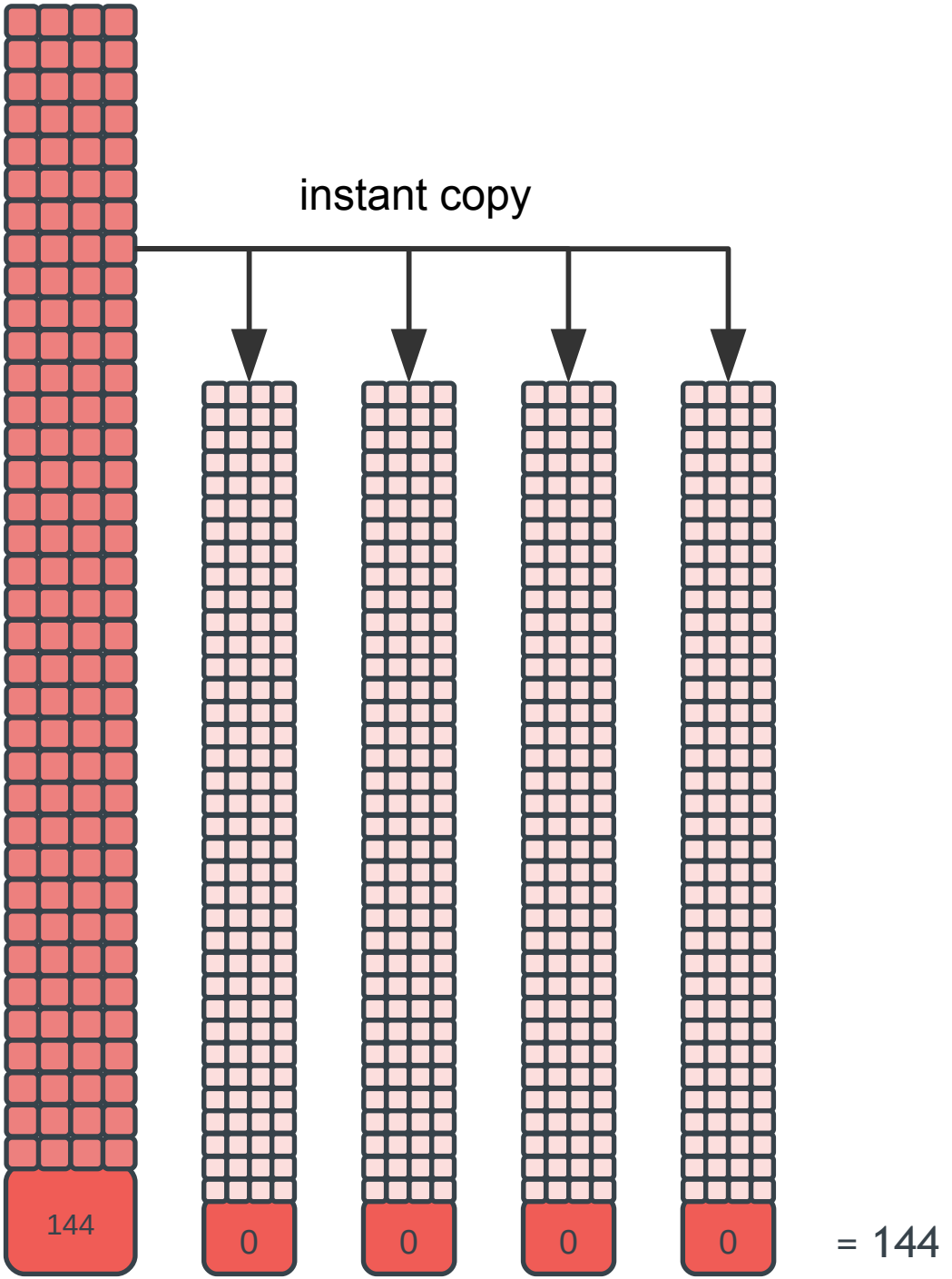
SPIN UP

THOUSANDS OF VMs

INSTANTLY

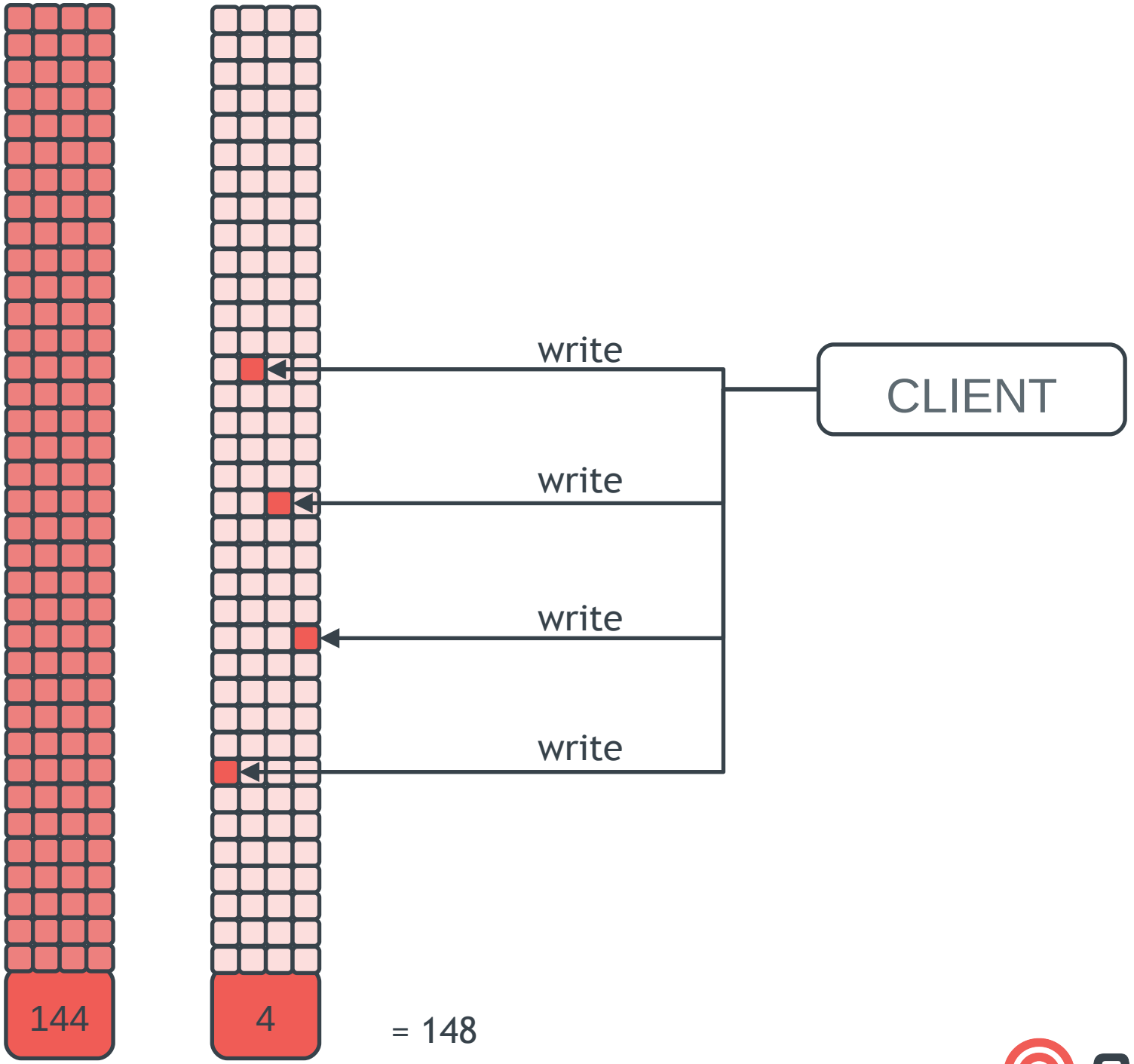
AND

EFFICIENTLY?

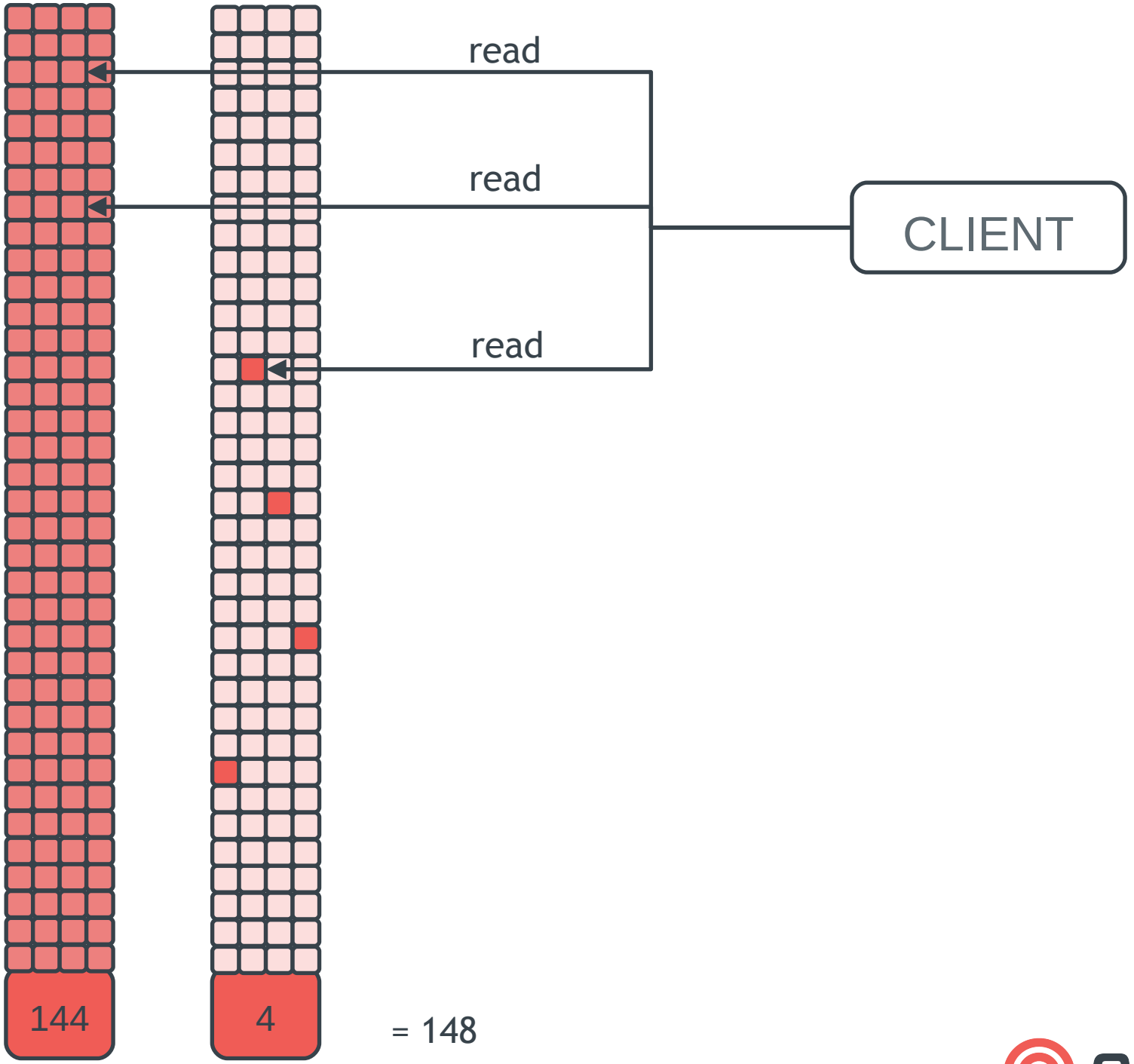


= 144

ceph



ceph



ceph

current RBD integration

- native Linux kernel support
  - /dev/rbd0, /dev/rbd/<poolname>/<imagename>
- librbd
  - user-level library
- Qemu/KVM
  - links to librbd user-level library
- libvirt
  - librbd-based storage pool
  - understands RBD images
  - can only start KVM VMs... :-(

ceph

what about Xen?

- Linux kernel driver (i.e. /dev/rbd0)
  - easy fit into existing stacks
  - works today
  - need recent Linux kernel for dom0
- blktap
  - generic kernel driver, userland process
  - easy integration with librbd
  - more featureful (cloning, caching), maybe faster
  - doesn't exist yet!
- rbd-fuse

ceph

libvirt

- CloudStack, OpenStack
- libvirt understands rbd images, storage pools
  - xml specifies cluster, pool, image name, auth
- currently only usable with KVM
- could configure /dev/rbd devices for VMs

librbd

- management
  - create, destroy, list, describe images
  - resize, snapshot, clone
- I/O
  - open, read, write, discard, close
- C, C++, Python bindings

RBD roadmap

- locking
  - fence failed VM hosts
- clone performance
- KSM (kernel same-page merging) hints
- caching
  - improved librbd caching
  - kernel RBD + bcache to local SSD/disk

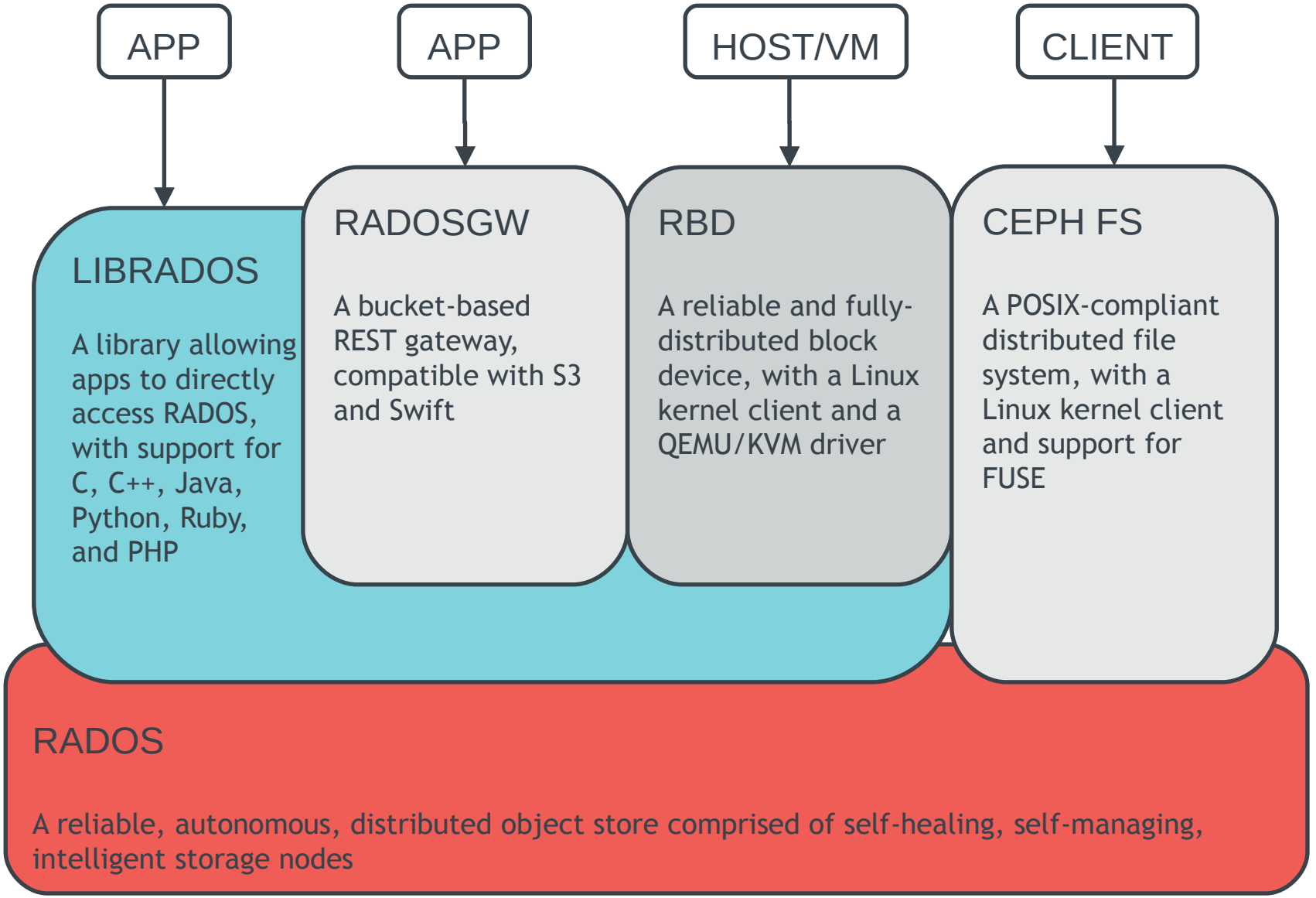
ceph

why

- limited options for scalable open source storage
- proprietary solutions
  - marry hardware and software
  - expensive
  - don't scale (out)
- industry needs to change

who we are

- Ceph created at UC Santa Cruz (2007)
- supported by DreamHost (2008-2011)
- Inktank (2012)
- growing user and developer community
- we are hiring
  - C/C++/Python developers
  - sysadmins, testing engineers
  - Los Angeles, San Francisco, Sunnyvale, remote


APP
APP
HOST/VM
CLIENT
LIBRADOS
A library allowing apps to directly access RADOS, with support for C, C++, Java, Python, Ruby, and PHP
RADOSGW
A bucket-based REST gateway, compatible with S3 and Swift
RBD
A reliable and fully-distributed block device, with a Linux kernel client and a QEMU/KVM driver
CEPH FS
A POSIX-compliant distributed file system, with a Linux kernel client and support for FUSE
RADOS
A reliable, autonomous, distributed object store comprised of self-healing, self-managing, intelligent storage nodes

ceph