ceph

THE STATE OF CEPH, MANILA, AND CONTAINERS IN OPENSTACK

SAGE WEIL

OPENSTACK SUMMIT TOKYO - 2015.10.28

OUTLINE

- CephFS
- CephFS status update
- Current Manila landscape
- CephFS native driver
- Better FS plumbing to VMs
- Manila vs Nova responsibilities
- Manila and containers
- Summary

CEPHFS

WHY USE FILE IN THE CLOUD?

Why file?

- File-based applications aren't going away
  - POSIX is lingua-franca
- Interoperability with other storage systems and data sets
- Container “volumes” are file systems
  - probably just directories
- Permissions and directories are useful concepts

Why not block?

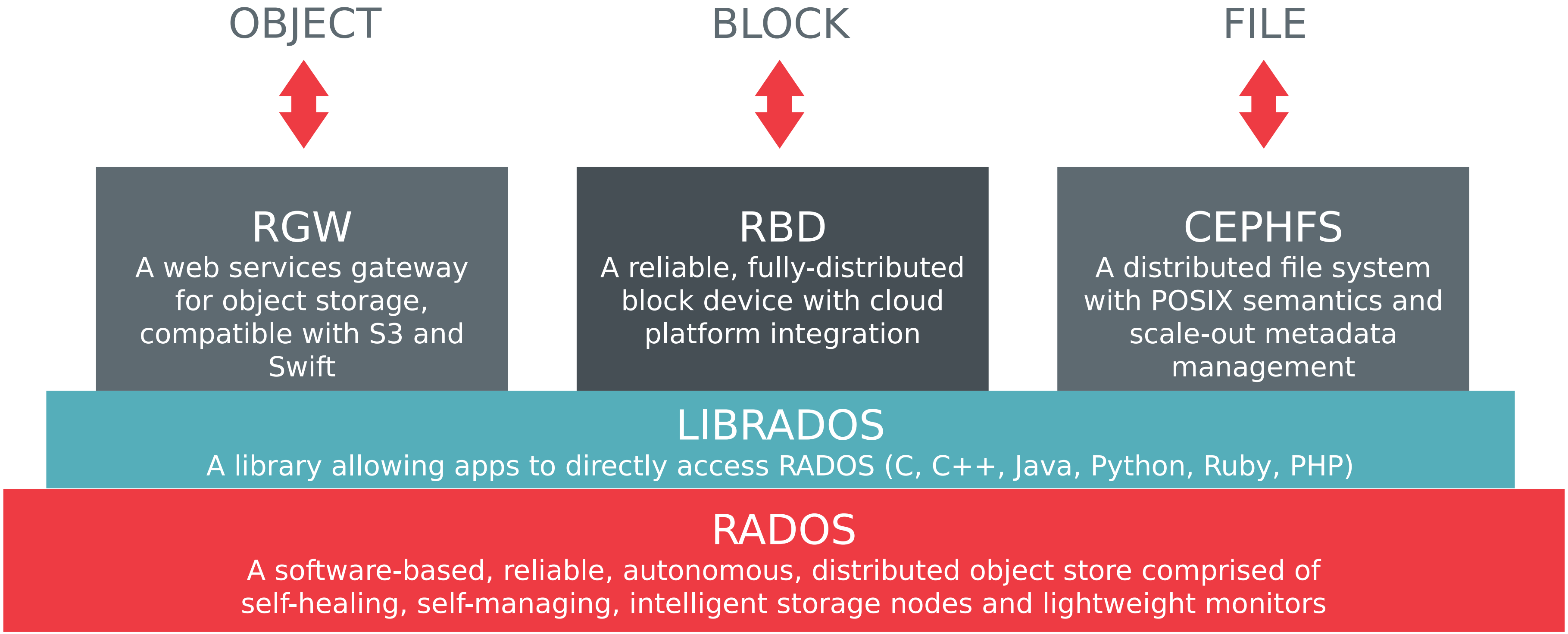
- Block is not useful for sharing data between hosts
  - ext4, XFS, etc assume exclusive access
- Block devices are not very elastic
  - File volumes can grow or shrink without administrative resizing

WHY CEPH?

- All components scale horizontally
- No single point of failure
- Hardware agnostic, commodity hardware
- Self-manage whenever possible
- Open source (LGPL)

- Move beyond legacy approaches
  - client/cluster instead of client/server
  - avoid ad hoc approaches HA

CEPH COMPONENTS



OBJECT

BLOCK

FILE

RGW

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

RBD

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

CEPHFS

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

LIBRADOS

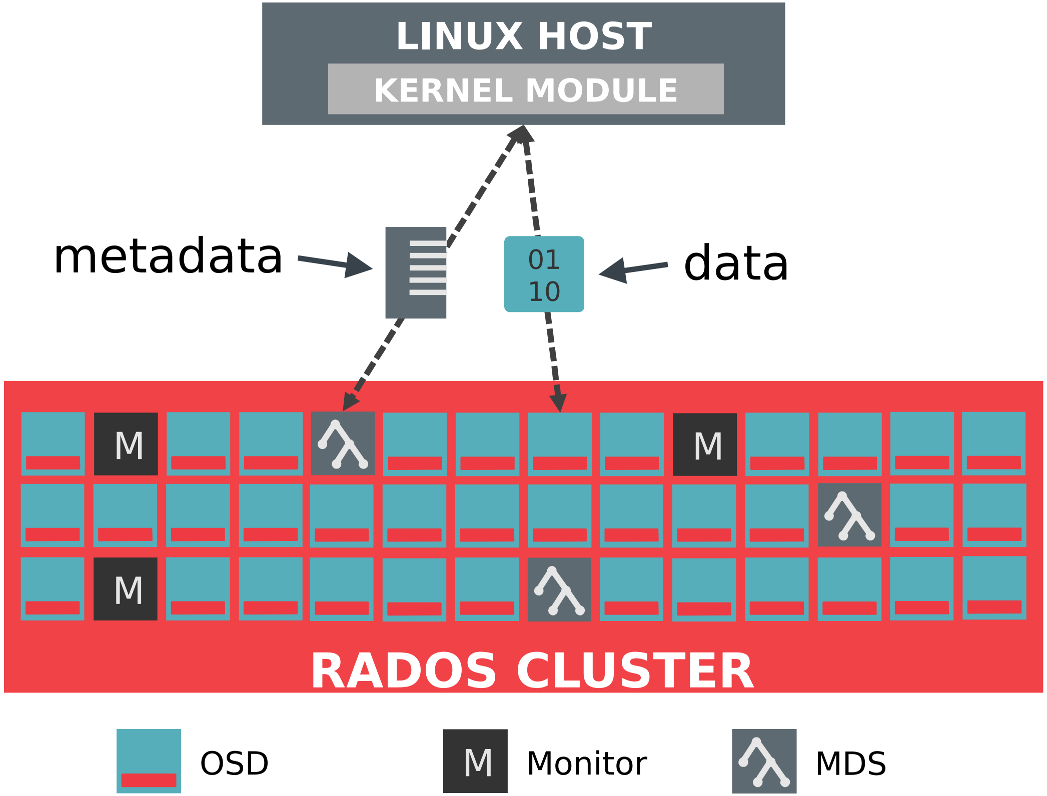
A library allowing apps to directly access RADOS (C, C++, Java, Python, Ruby, PHP)

RADOS

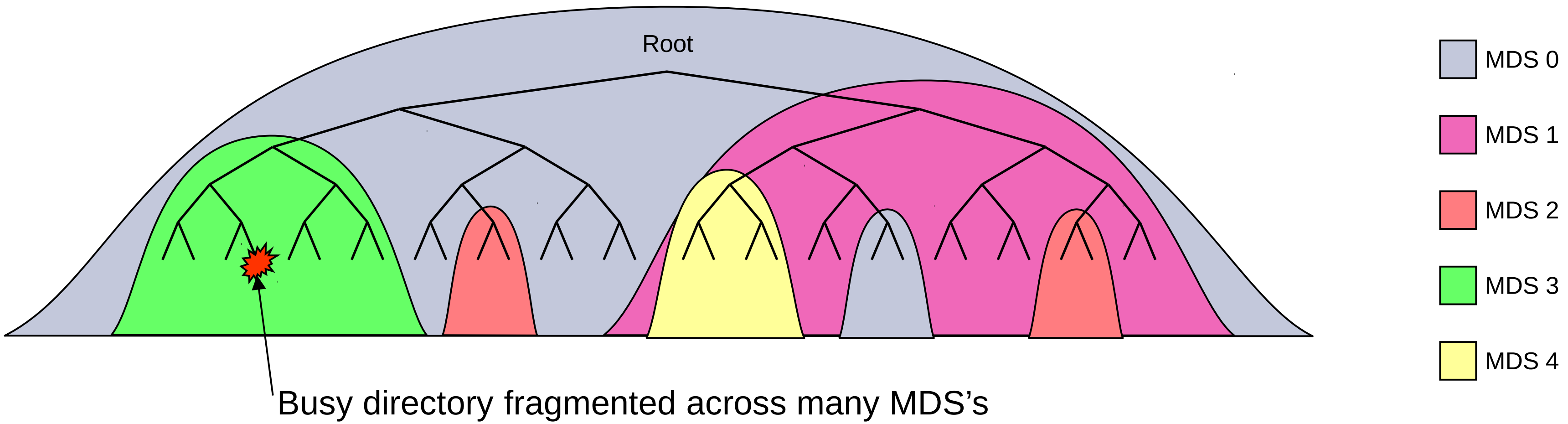
A software-based, reliable, autonomous, distributed object store comprised of self-healing, self-managing, intelligent storage nodes and lightweight monitors

CEPHFS DISTRIBUTED FILE SYSTEM

- CephFS
  - scalable data: files are stored directly in RADOS
  - scalable metadata: cluster of metadata servers (MDS)
  - POSIX: drop-in replacement for any local or network file system
- Multiple clients
  - Linux kernel
  - ceph-fuse
  - libcephfs.so
    - Samba, Ganesha, Hadoop



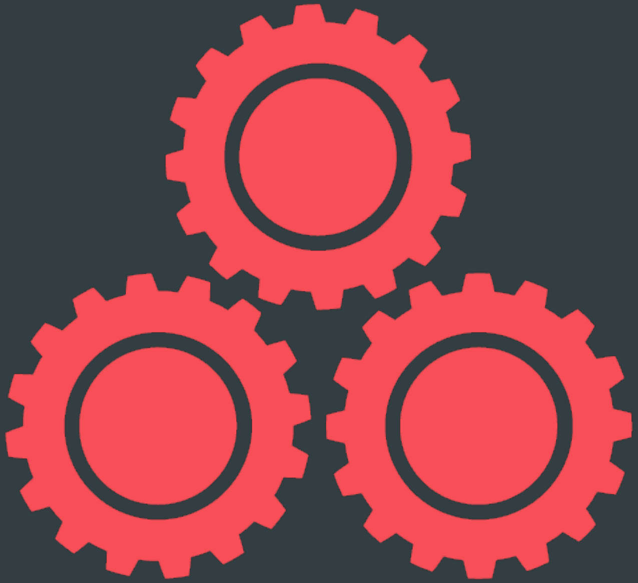
CEPHFS DYNAMIC SUBTREE PARTITIONING



- Scalable
  - Arbitrarily partition hierarchy
  - 10s to 100s of MDSs
- Adaptive
  - Move load from busy to idle servers
  - Replicate hot metadata on multiple nodes

OTHER GOODIES

- Strongly consistent / coherent client caches
- Recursive accounting
  - directory file size is amount of data stored
- Snapshots
  - on any directory
- Directory quotas (libcephfs/ceph-fuse only)
  - limit by bytes or file count
- xattrs
- ACLs
- Client-side persistent cache (Kernel client only)



CEPHFS STATUS UPDATE

ROAD TO PRODUCTION

- Focus on resilience
  - handle errors gracefully
  - detect and report issues
  - provide recovery tools
- Achieve this first with a single-MDS configuration
- CephFS as dog food
  - use CephFS internally to run our QA infrastructure
  - have found (and fixed) several hard to reproduce client bugs

- “Production-ready” CephFS with Jewel release (Q1 2016)

WHAT IS NEW, WORK IN PROGRESS

- Improved health checks for diagnosing problems
  - misbehaving clients, OSDs
- Diagnostic tools
  - visibility into MDS request processing
  - client session metadata (who is mounting what from where)
- Full space handling
- Client management
  - evict misbehaving or dead clients
- Continuous verification
  - online scrubbing of metadata

FSCK AND REPAIR

- Repair tools
  - loss of data objects (which files are damaged)
  - loss (or corruption) of metadata objects (which subtrees are damaged)
- cephfs-journal-tool
  - disaster recovery for damaged MDS journal
  - repair damaged journal
  - recover metadata from damaged or partial journal
- cephfs-table-tool
  - adjust/repair/reset session, inode, snap metadata
- cephfs-data-scan
  - rebuild metadata (directory hierarchy) from data objects (disaster recovery)

ACCESS CONTROL

- Path-based
  - restrict client mount to a subdirectory (e.g., /volumes/foo or /home/user)
  - implemented in MDS
  - integration with RADOS namespaces is WIP (targeting Jewel)
  - (Jashan from GSoC)
- User-based
  - mount file system with as single user (or small set of users)
  - UID and GID based
  - implement Unix-like permission checks at the MDS
  - eventual integration with external auth/auth frameworks (e.g., Kerberos/AD)
  - (Nishtha from Outreachy)

THE CURRENT MANILA LANDSCAPE

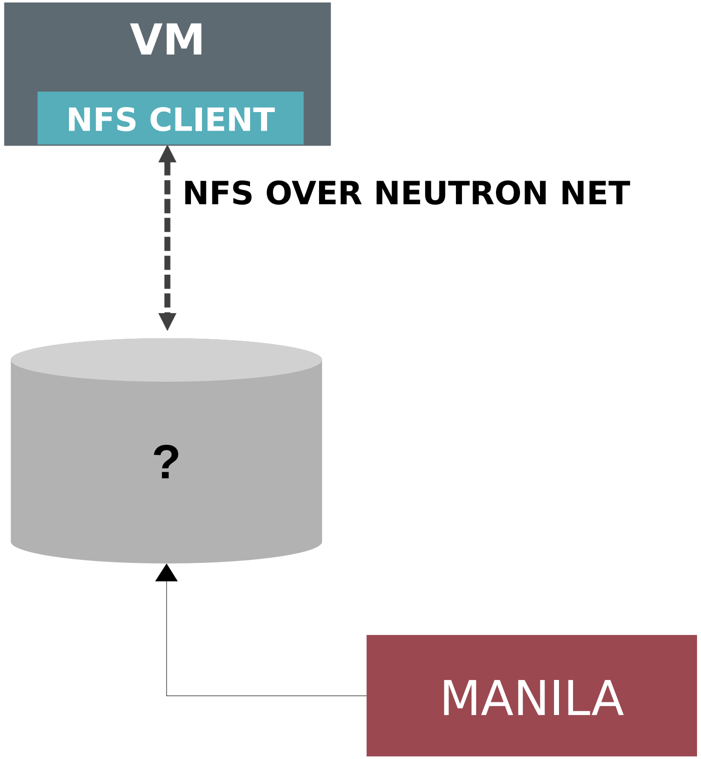
MANILA FILE STORAGE

MANILA

- Manila manages file volumes (“shares”)
  - create/delete, share/unshare
  - file server network connectivity
  - snapshot management
- Caveats, awkward bits
  - Manila also manages (only) part of the connectivity problem
    - somewhat limited view of options (network file protocols only)
    - manages “share networks” via Neutron
  - User has responsibility for the “last mile”
    - user must attach guest to share network
    - user must mount the share (mount -t …)
    - mount mechanism varies with storage type and/or hypervisor (NFS or CIFS)

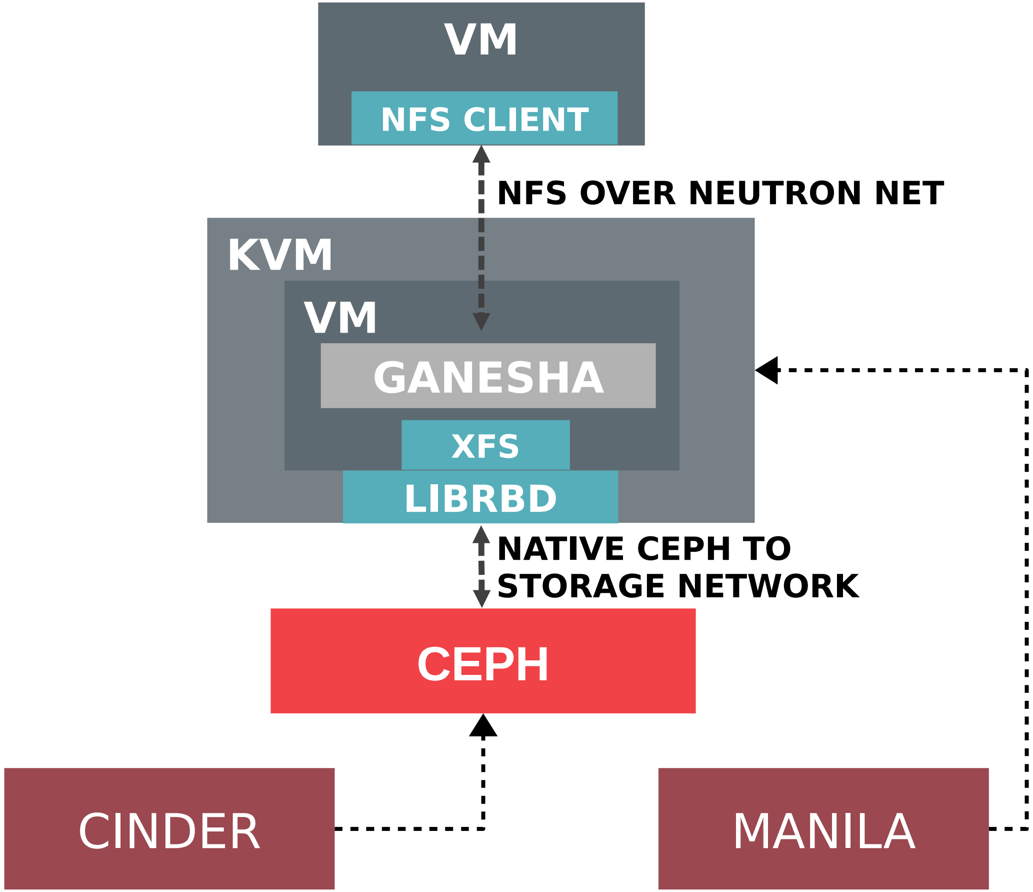
APPLIANCE DRIVERS

- Appliance drivers
  - tell an appliance to export NFS to guest IP
  - map appliance IP into tenant network (Neutron)
  - boring (closed, proprietary, expensive, etc.)
- Status
  - several drivers from usual suspects
  - security punted to vendor



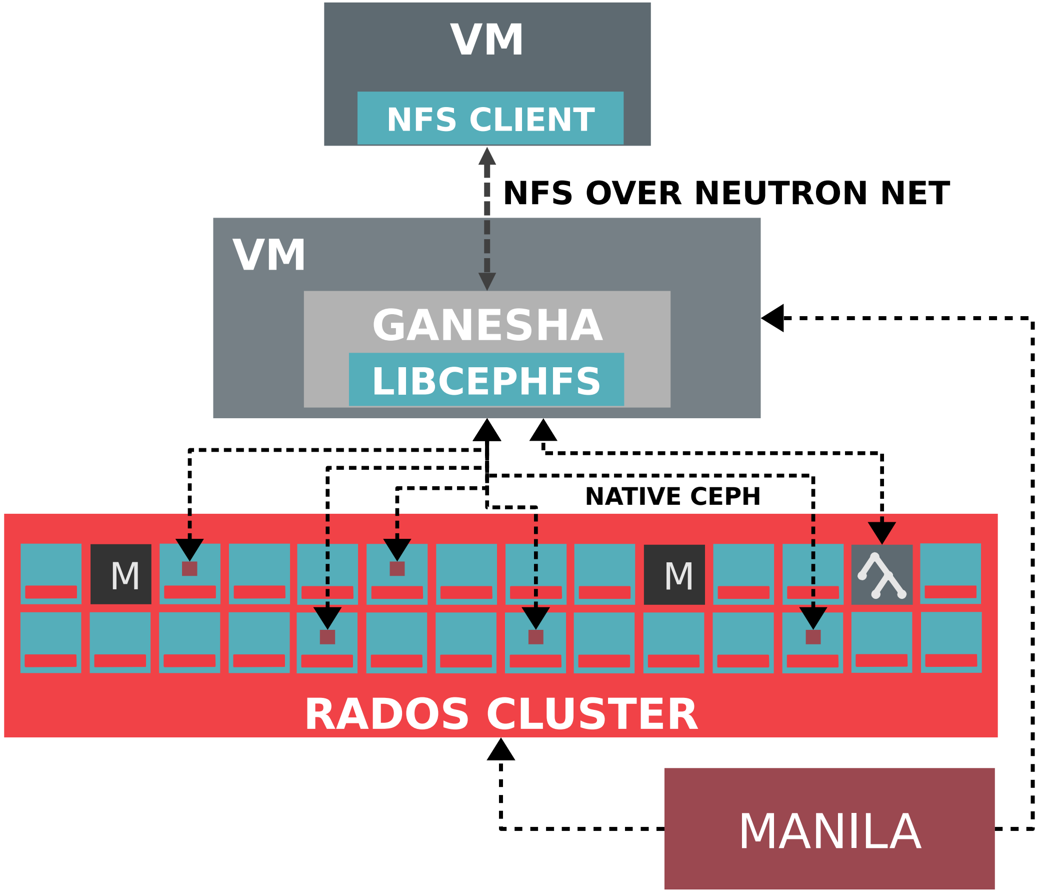
GENERIC SHARE DRIVER

- Model
  - Cinder volume attached to service VM
  - local file system (XFS, ext4, btrfs, ...)
  - Ganesha NFS server
  - Neutron network shared with tenant
- Pros
  - built from existing components
  - tenant isolation, security
- Cons
  - extra hop → higher latency
  - service VM consumes resources
  - service VM is SPoF
- Status
  - reference driver



GANESHA + LIBCEPHFS

- Model
  - existing Ganesha driver toolkit, currently used by GlusterFS
  - Ganesha's libcephfs FSAL
- Pros
  - simple, existing model
  - security
- Cons
  - extra hop → higher latency
  - service VM is SpoF
  - service VM consumes resources
- Status
  - Manila Ganesha toolkit exists
  - used for GlusterFS
  - not yet integrated with CephFS



THE PROBLEM WITH SERVICE VMS

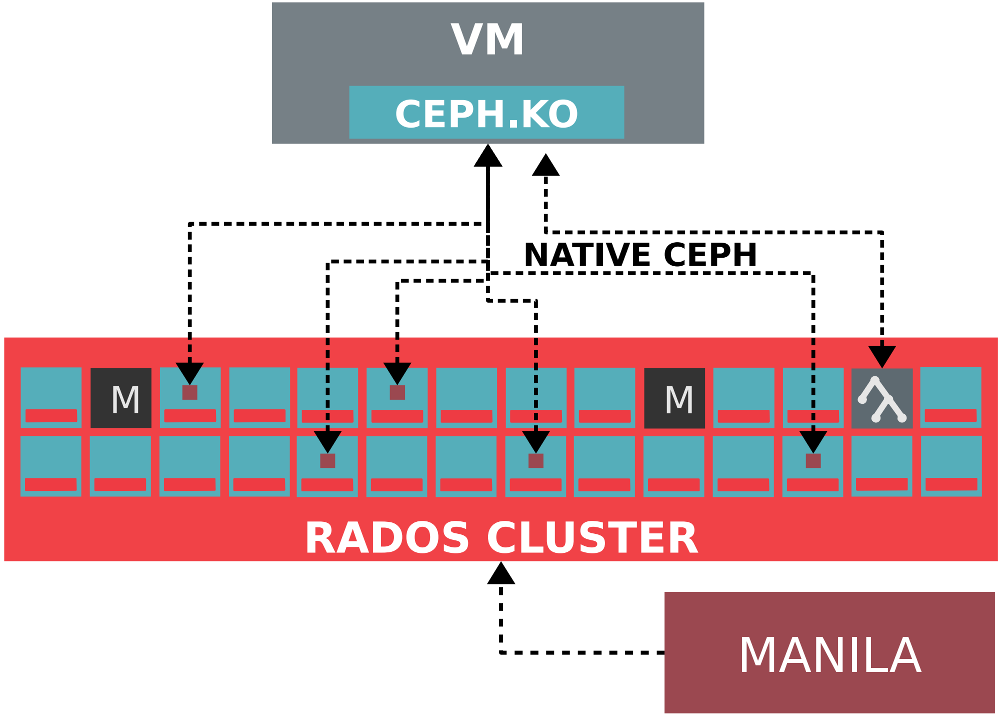
- Architecture is limited
  - slow: extra hop
  - expensive: extra VM
- Current implementation is not highly-available
  - need service monitoring, failover
  - possibly load balancing
  - Manila code assumes a single service endpoint/proxy

- It's a big TODO list. Is it the right end point?

NATIVE CEPHFS MANILA DRIVER

CEPH NATIVE DRIVER

- Model
  - allow tenant access to storage network
  - mount CephFS directly from tenant VM
- Pros
  - best performance
  - access to full CephFS feature set
  - simple
- Cons
  - guest must have modern distro/kernel
  - exposes tenant to Ceph cluster
  - networking currently left to user
  - must deliver mount secret to client

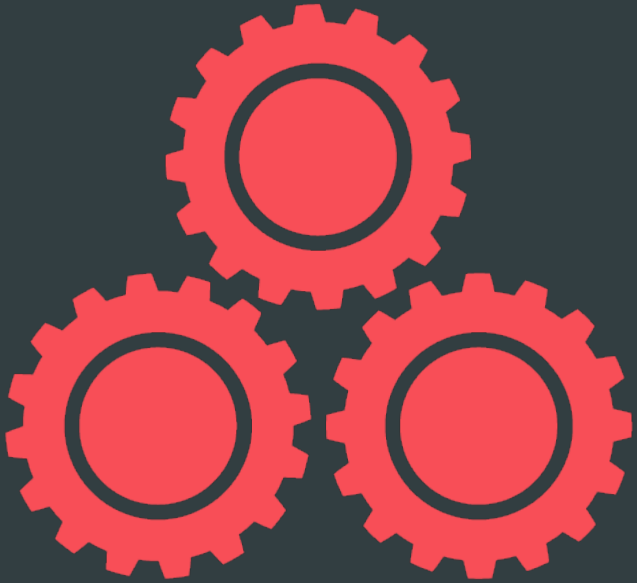


CEPHFS-VOLUME-MANAGER.PY

- cephfs-volume-manager.py → libcephfs.py → libcephfs.so → CephFS
  - will be packaged as part of Ceph (with python-cephfs)
- Manila volumes/shares and consistency groups are just CephFS directories
  - e.g., /manila/$cg/$volume
- Capture useful CephFS volume management tasks
  - create – mkdir /manila/$cg/$volume
  - delete (async) – mv /manila/$cg/$volume /manila/.trash
  - snapshot volume – mkdir /manila/$cg/$volume/.snapshot/$snapname
  - snapshot consistency group – mkdir /manila/$cg/.snapshot/$snapname
  - promote snapshot to new volume
    - read/write – cp -r ...
    - read only – ln -s /manila/$cg/$vol/.snapshot/$snap /manila/$cg/$newvol
- Result is very simple Manila driver
  - ~250 lines of code for native driver

SECURITY

- Tenant has access to the storage network
  - Ceph and CephFS are responsible for security isolation between tenants
- Client *authentication* has been there for years
  - modeled after Kerberos (mutual client/server authentication)
- New CephFS path-based *authorization*
  - new in MDS. Now upstream
  - missing CephFS support for rados namespaces
    - needed to restrict client access to CephFS objects using librados API
  - will be in Jewel (Q1 2016)
- Is that enough?
  - Ceph's security is the only barrier
  - DoS potential against cluster
  - it depends on the environment...



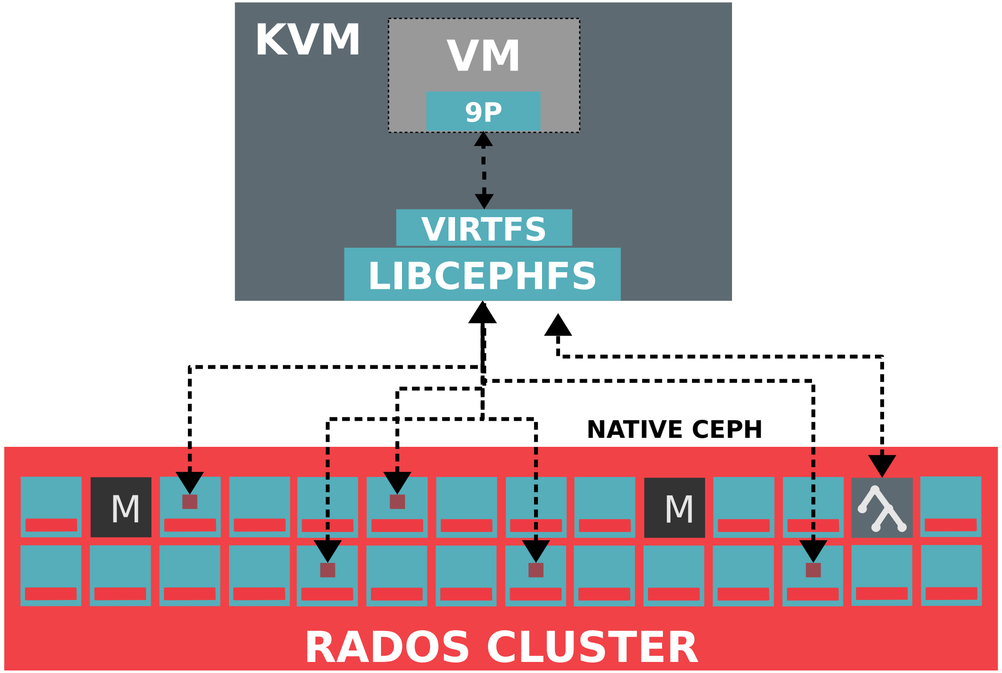
BETTER FS PLUMBING

WE WANT

- Better security
  - ...like we get with block storage
- Simplicity of configuration and deployment
  - ...like with Qemu and librbd
- Good performance
  - ...like with CephFS native

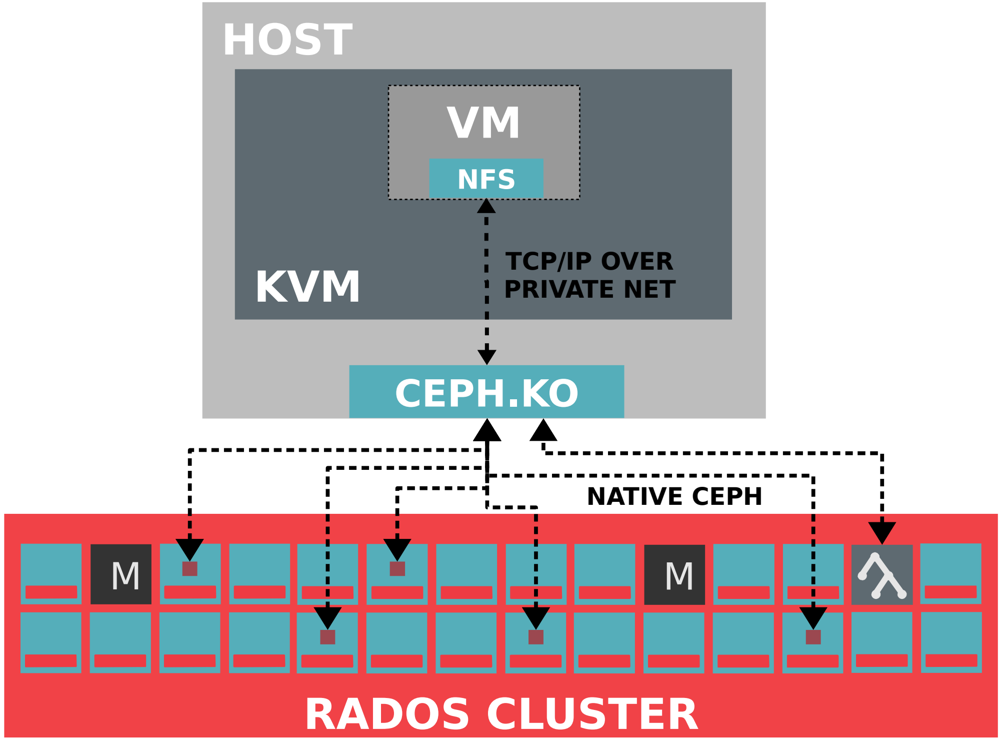
KVM + 9P/VIRTFS + LIBCEPHFS.SO

- Model
  - link libcephfs.so into qemu virtfs layer
  - guest OS (Linux) mounts via 9P-2000.L
- Pros
  - security: tenant remains isolated from storage net + locked inside a directory
  - extremely simple deployment
- Cons
  - requires (modern) Linux guests
  - not supported on some distros
    - 9p (kernel) and virtfs (qemu) code quality
  - 9P isn't the great file protocol
- Status
  - Prototype from Jevon Qiao, Haomai Wang, et al
    - Qemu virtfs + libcephfs
    - Manila driver + Nova mods



KVM + NFS + NFSD/GANESHA + CEPHFS

- Model
  - mount CephFS on host
    - or Ganesha + libcephfs on host
  - export NFS to guest over private net
- Pros
  - security: tenant remains isolated from storage net + locked inside a directory
  - NFS is well supported everywhere
  - reliable: same HW failure domain as guest
  - works for any FS, not just CephFS
- Cons
  - NFS has weak caching consistency
  - protocol translation will slow us down some
  - awkward and/or insecure networking...



NFS TO HOST: PROBLEMS WITH TCP/IP

- Slightly awkward networking
  - add dedicated network device to VM
  - configure local subnet and assign IPs on host and guest
  - configure NFS export on hypervisor
  - mount export from the VM
- Tricky to automate special-purpose network interfaces
- Guest networking infrastructure can disrupt file sharing
  - firewalld
  - networking restart
  - “What is this weird network and interface doing here?”
- Other services on host may inadvertently be exposed to guest
  - anything binding to INADDR_ANY (e.g., sshd)

AF_VSOCK

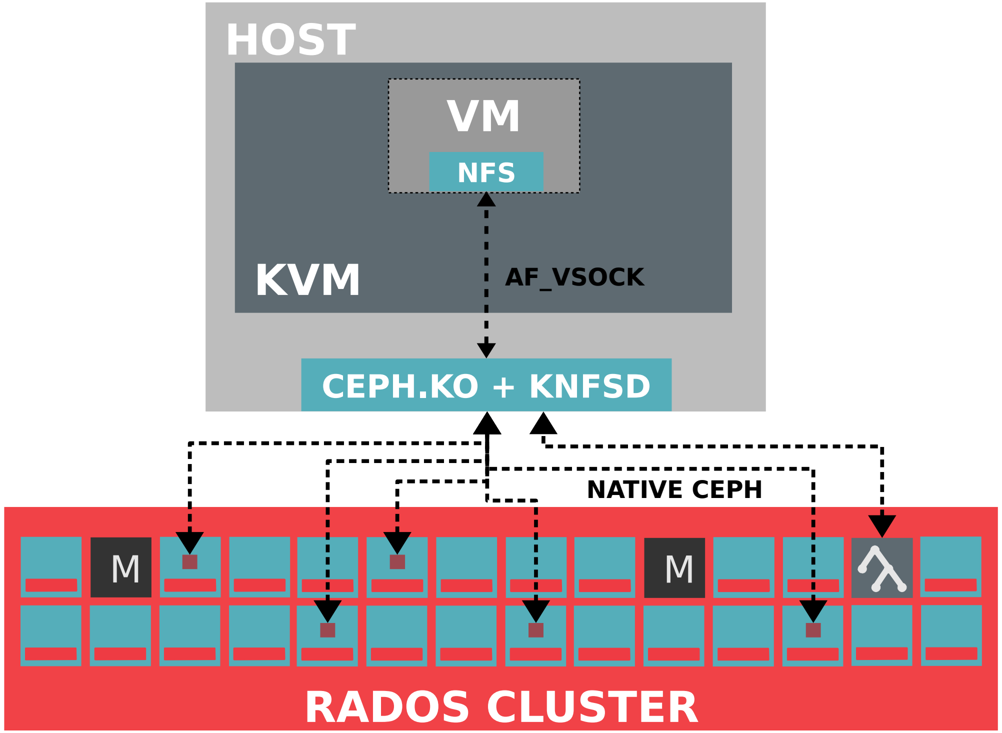
- VMware vSockets: a new(-ish) address family / socket type
  - designed for communication between VMs and hosts
  - stream-based or connectionless datagrams (just like IP)
  - address is a simple integer (e.g., vsock:2)
  - supported in Linux kernel since v3.9 (2013)
- Zero configuration simplicity
  - hypervisor is always address vsock:1
  - hypervisor assigns an address >1 to each VM

NFS TO HOST: VSOCK

- NFS v4.1 only
  - older NFS versions have awkward legacy connectivity/addressing requirements (e.g., lockd)
  - v4.1 consolidates protocol into a single connection
- Easy to support
  - mostly boilerplate to add new address type, as with IPv6 (e.g., parsing)
- Linux kernel NFS client and server
  - patches from Stefan Hajnoczi are under review
- Ganesha
  - patches from Matt Benjamin are under review
- nfs-utils
  - patches from Matt Benjamin are pending review

KVM + NFS (VSOCK) + NFSD + CEPHFS.KO

- Model
  - mount CephFS on host, knfsd
    - or Ganesha + libcephfs on host
  - export to VM's VSOCK address
- Pros
  - NFSv4.1 is well supported
  - security is better...
  - simpler configuration...
  - more reliable...
- Cons
  - VSOCK support for Qemu and NFS is shiny and new



WE LIKE THE VSOCK-BASED MODEL

- Security
  - tenant remains isolated from storage network
  - no shared IP network between guest and host
    - avoid INADDR_ANY problem (e.g., by sshd on host)
- Simplicity
  - no network configuration beyond VM VSOCK address assignment (on host)
  - treats VM as a black box
  - no software-defined networking
- Reliability
  - no gateway in a separate hardware failure domain
  - fewer network traversals
- Performance
  - clear win over a service VM
  - possible win over TCP/IP to host (but not currently optimized with this in mind!)

VSOCK CHALLENGES

- New hotness
  - need to get code upstream and intro supported distros/products
  - Qemu, Linux kernel, Ganesha, nfs-utils
- Host configuration
  - someone needs to assign VSOCK addresses to VMs
  - someone needs to mount CephFS (or other FS) on the host and reexport NFS to the guest
- User experience and the last mile
  - How does a consumer of the Manila API know how to mount this thing?
  - Do they need intimate knowledge of which Manila driver is in use, and what attachment mechanism is supported by this particular OpenStack instance?
  - Can they choose?

MANILA VS NOVA RESPONSIBILITIES

MANILA VS NOVA

- Manila manages shares/volumes
- Nova manages the VMs

- Cinder manages block volumes
- Nova manages VMs
- Nova attaches Cinder volumes to VMs
  - mechanism is dependent on the Nova driver (KVM vs Xen vs lxd vs ...)

MANILA

vs

NOVA

NOVA: ATTACH/DETACH FS API

- Attach or detach a file system
  - hypervisor mediates access to Manila shares/volumes
  - networking?
    - attach to Neutron network
    - assign VSOCK address
  - gateway/proxy?
    - knfds or Ganesha
  - containers?
- Fetch access metadata (e.g., mount command inputs)
  - mount protocol and options depend on Nova instance type **and** share type
- Now Nova...
  - can reattach after reboot
  - manage live migration

MANILA

vs

NOVA

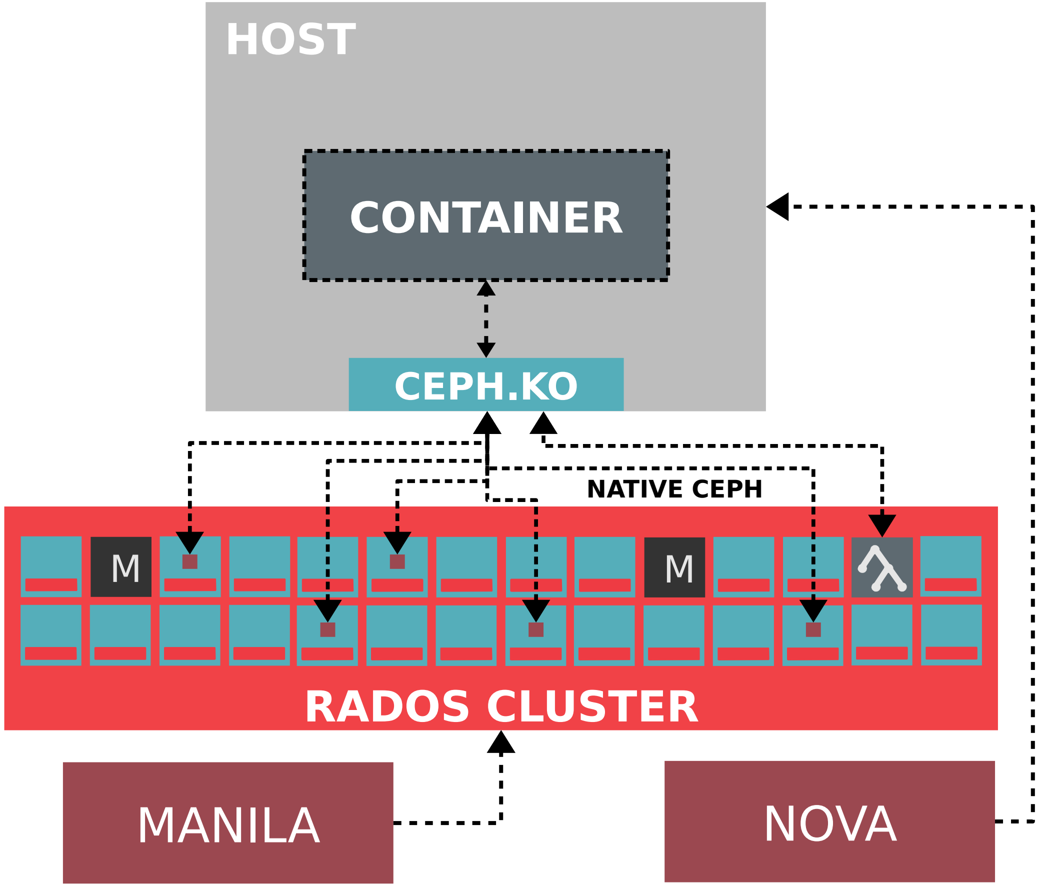
NOVA: ATTACH/DETACH FS

FS access mode	Meaning of attach/detach	Meaning of access metadata
KVM, NFS from guest (e.g., to NetApp)	Attach guest to Manila share's network	Typical NFS mount command: mount -t nfs $filervip:/ ...
KVM, VSOCK, Ganesha, libcephfs/libgfapi	Write share definition to local Ganesha config file for guest's VSOCK addr. Start Ganesha.	NFS VSOCK mount command: mount -t nfs vsock://1/ ...
KVM, VSOCK, knfsd, cephfs.ko mount	Mount Cephfs. Write share definition to /etc/exports for guest's VSOCK addr. exportfs -a	NFS VSOCK mount command: mount -t nfs vsock://1/ ...
KVM, NFS to generic share driver	Attach guest to Manila share's network	NFS IP mount command: mount -t nfs $filerip:/ ...
KVM, NFS to Ganesha service VM	Attach guest to Manila share's network	NFS IP mount command: mount -t nfs $filerip:/ ...
KVM or Ironic, native CephFS	No-op (or, attach guest to storage network)	CephFS mount requires a secret: mount -t ceph $monip:/ -o secret=X ...
Nova container (lxc, lxd)	Mount remote fs on host; mount –bind share to guests /dev/manila/$shareid	Bind mount to desired location: mount –bind /dev/manila/$shareid ...

WHAT ABOUT CONTAINERS

(LXC, LXD) + CEPHFS.KO

- Model
  - host mounts CephFS (or whatever) directly
  - mount --bind share into container namespace (/dev/manila/$shareid)
  - user does mount --bind to final location
- Pros
  - best performance
  - full CephFS semantics
- Cons
  - rely on container for security
  - need Nova attach/detach API



SUMMARY

- Ceph native driver should land soon
  - and Ceph Jewel (Q1 2016) will have production-ready CephFS!
- Current Manila models are appliance-centric or limited
- NFS over VSOCK to the host is promising
  - simplicity, reliability, security, performance
  - either kernel NFS server or Ganesha
- We need to sort out the Nova vs Manila interaction
  - Nova APIs would help enable
    - non-KVM users for Manila (**containers**, Ironic)
    - NFS over VSOCK to a host gateway

THANK YOU!

Sage Weil

CEPH PRINCIPAL ARCHITECT

✉ sage@redhat.com

@liewegas

FOR MORE INFORMATION

- Ceph
  - http://ceph.com
  - http://github.com/ceph
  - http://tracker.ceph.com
- Mailing lists
  - ceph-users@ceph.com
  - ceph-devel@vger.kernel.org
- irc.oftc.net
  - #ceph
  - #ceph-devel
- Twitter
  - @ceph

- Qemu + libcephfs, w/ Nova and Manila support
  - https://github.com/JevonQ/qemu/commit/3c5d09149b59735905388ed51861c018c7737e7e
  - https://github.com/yuyuyu101/nova/tree/bp/manila-virtfs-support
  - https://github.com/yuyuyu101/nova/tree/bp/manila-virtfs-support
- Qemu virtio-vsock
  - https://lwn.net/Articles/646365/
  - https://github.com/stefanha/qemu/commits/vsock
- Linux NFS client/server VSOCK support
  - https://github.com/stefanha/linux/commits/vsock-nfs
  - https://copr.fedoraproject.org/coprs/jspray/vsock-nfs/builds/
- Ganesha VSOCK support
  - https://github.com/linuxbox2/nfs-ganesha/tree/vsock
- Ceph native manila driver
  - https://github.com/jcsp/manila/commits/ceph
- cephfs-volume-manager.py
  - https://github.com/ceph/ceph/pull/6205