OpenStack File Shares Integration Scenarios

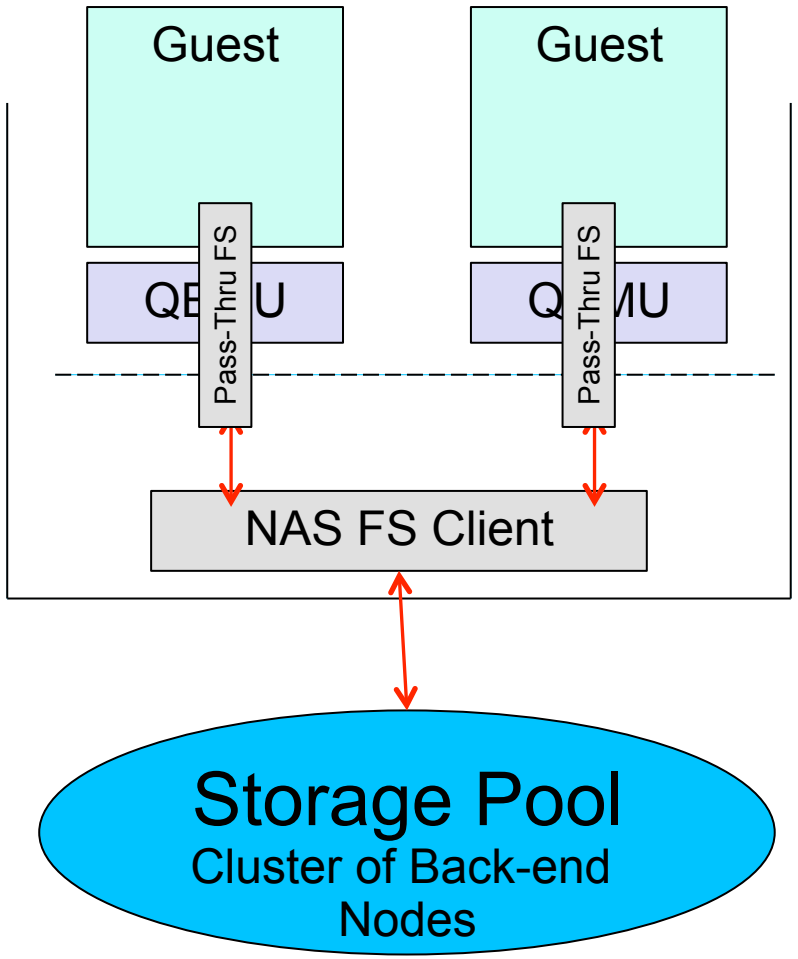
Software-defined-Storage Use-cases

B. Bellur, D. Williams
Red Hat
16 July 2013 (minor updates 8/22)

Software Defined Storage
Glusterfs as illustrative example

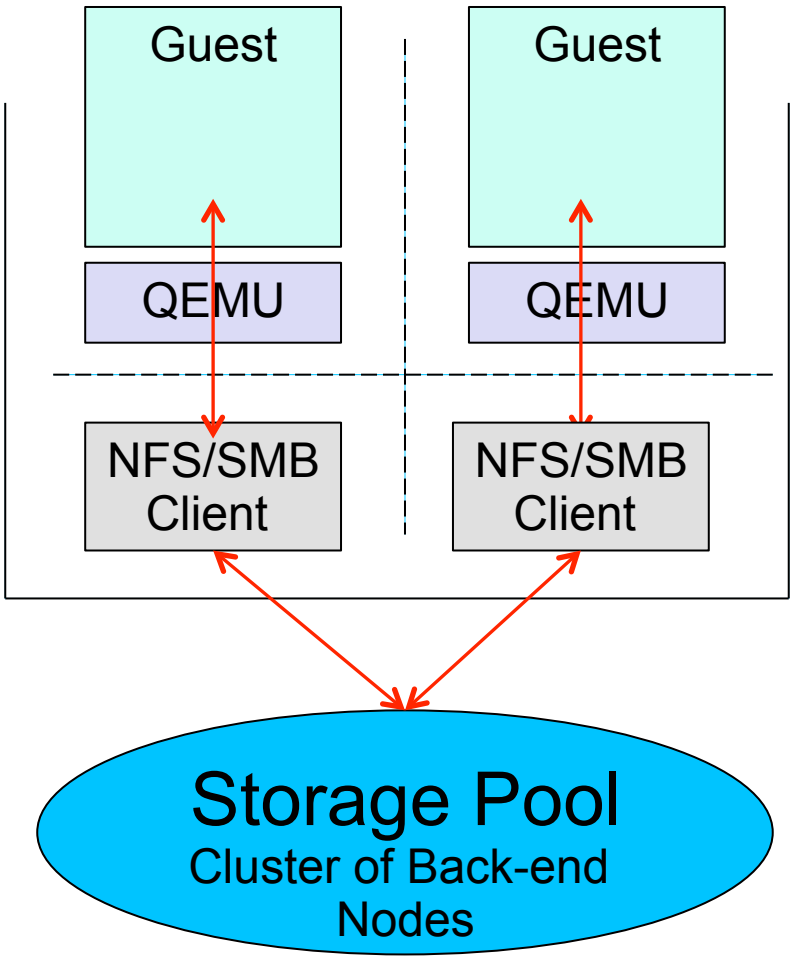
- Software stack federates pools of local storage into unified storage pool
  - Data Distribution: Ability to scale-out by adding individual local storage nodes
  - Data Availability: Data replication to survive failures at either the component or node level
- General Structure:
  - Front-end/Client-side: Exposes unified/available view to consuming applications
    - Native APIs: Libgfapi, Glusterfs FUSE
      - Ceph equivalents are librados & Linux Ceph provider
    - NFS, SMB: Access layer adapts native client (internal api) to standard protocols
  - Back-end/Server-side: Cluster of individual (discrete) data persistence pools (unit of capacity, units of failure.
  - Front-end and Back-end typically combined when delivering storage as a standalone pool (resource at the other end of the wire)

Hypervisor Mediated Storage Access via Pass-Through FileSystem



- VM host mounts shared filesystem
- Hypervisor exports subset of host-mounted to guest
  - Pass-through filesystem responsible for name-space mapping, security
  - Pass-through filesystem hypervisor-dependent
    - Vmware: VMFS
    - KVM: Virtfs (immature)
    - Xen: ?
- Issue (KVM):
  - VirtFS maturity. Currently not supported on RHEL/CENTOS
  - Virtfs client support for Windows

Hypervisor Mediated Storage Access NFS and SMB

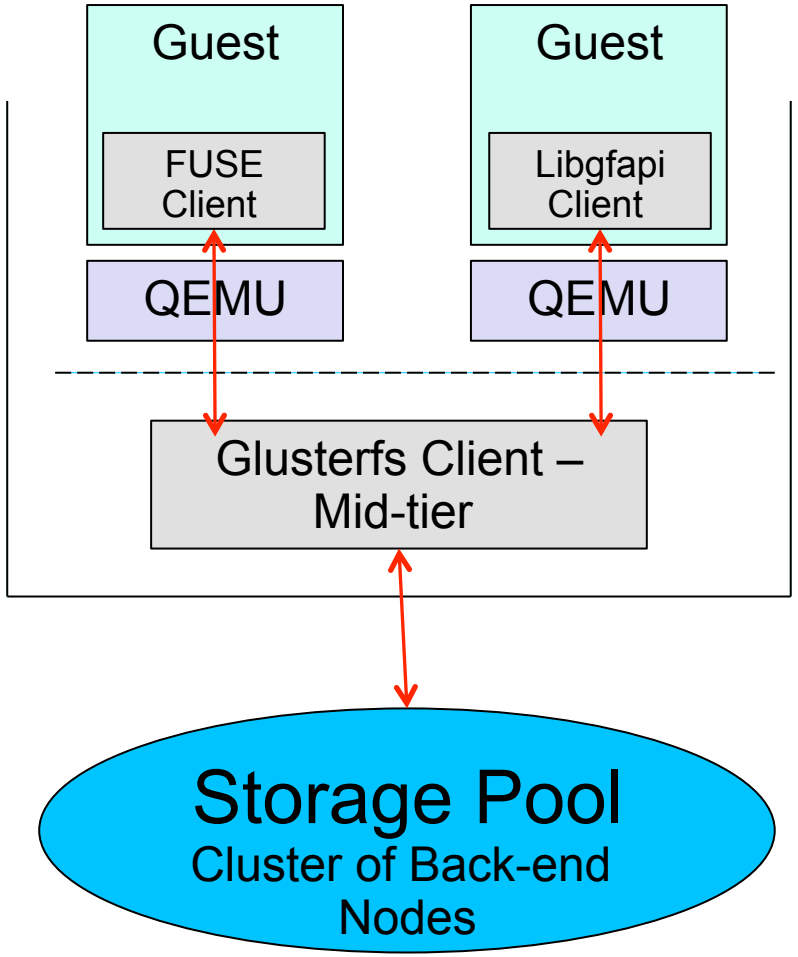


- NFS/SMB exposed to guests via per-guest host-resident client
  - Gluster NFS Client lives within netns partition for isolation
  - Dedicated point-to-point virt network for communication between client and guest
- Mapping of tenant virtual resource to provider
  - KeyStone cert with config tuple, one cert per virtual volume:
  - Identity of resource in tenant name space (ie: tenant NFS namespace)
  - Mapping to provider resource, such as Glusterfs volume containing virt file share
  - Security credentials restrict client to only access tenant's resources
  - Other mapping data as needed
- Issue: Mechanics of attach operation:
  - Fully storage provider managed (or)
  - Nova assisted with call backs for attach/detach

Why Not Just Put NFS/SMB in Storage pool, interconnect via Neutron or dedicated Network?

- Hypervisor-based guest may have data placement awareness, resulting in more efficient utilization of network for guest-to-storage pool traffic.
  - This is particularly for Big Data workloads such as Hadoop with Savanna
- Hyper-based client can have additional security or QoS features

Hypervisor Mediated Storage Access Native Protocol



- Glusterfs client split into:
  - Thin top-level client
  - Thick mid-tier client
- Thin top-level client lives in VM guest
  - Exposing FUSE or libgfapi.
  - View as untrusted from security perspective
- VM Host-resident Mid-tier client
  - Handles typical glusterfs processing (distribute, replicate),
  - Multiu-tenant aware for security
- Connectivity between Guests and Mid-tier client via
  - Dedicated point-to-point virtual networks
  - VSock

Virtual File-Share automation considerations

File Share Create/Delete/Snap operations can be implemented as a driver within the OpenStack FileShare service, similar to Cinder

Two alternatives for attachment:

- Have each nova guest contain a pluggable FileShare driver, callable from the master FileShare service
  - Performs hypervisor-side volume attach/detach operations
  - Enables/disables client node (e.g.: client-side Glusterfs NFS server and/or mid-tier client process
- Static configuration:
  - Network plumbing and VM host-side client process provisioned at time of Nova instance creation. Default is for client to not expose any file shares
  - Through GlusterFS CLI, provision client to present/unpresent specific file shares
  - Glusterfs FileShare plugin (or glusterfs itself) maintains mapping between clients and associated Nova instances.