OpenStack Cloud Storage

Sam Fineberg
HP Storage
What is OpenStack®

Free open source (Apache license) software governed by a non-profit foundation (corporation) with a mission to produce the ubiquitous Open Source Cloud Computing platform that will meet the needs of public and private clouds regardless of size, by being simple to implement and massively scalable.

- **Massively scalable** cloud operating system that controls large pools of **compute**, **storage**, and **networking** resources
- **Community open source** with contributions from **1000+ developers** and **180+ participating organizations**
- **Open** web-based API **Programmatic Infrastructure** as a Service
- **Plug-in architecture**; allows different hypervisors, block storage systems, network implementations, hardware agnostic, etc.
What is OpenStack®

A series of interrelated projects that control pools of compute, storage, and networking infrastructure exposed as a consistent and open layer (API) for a heterogeneous Infrastructure as a Service (IaaS) environment.
OpenStack® programs

13 packages with 200+ configuration items

- Identity (Keystone)
- Object Storage (Swift)
- Dashboard (Horizon)
- Compute (Nova)
- Network (Neutron)
- Block Storage (Cinder)
- Orchestration (Heat)
- Usage Metrics (Ceilometer)
- Bare metal (Ironic)
- Relational DB (Trove)

Image source: http://docs.openstack.org
OpenStack® releases

6-Month Release Cycles

• Spring and fall releases for predictable availability
• Release names use alphabetic naming

Planning
• Developers plan the next release at the Design Summit
• Sessions are selected by projects leads and are generally driven by blueprint topics
• Project Technical Leads (PTLs) accept a number of blueprints into the release plan

Development & Testing
• Milestone iterations (commonly 5 weeks) are defined and followed
• Development/documentation occurs per the blueprints and plan
• End of cycle testing with a defined Release Criteria process

Release
• Release Candidate is made available
• Next release is open for development
• After hardening, release occurs
Block Storage
• Generally SCSI protocol based, organized by LUNs
• Boot volumes for VMs
• Ephemeral vs. Persistent
• Not directly consumed by applications, usually used to hold a filesystem
• Low level storage abstraction upon which file and object storage is built

File Storage
• Files organized in directory hierarchy and accessed by pathname
• File-based NAS protocols like NFS and CIFS
• Rich and complex application support: random access, in-place file updates, locking, etc.

Object Storage
• Efficient flat namespace: objects organized by accounts, containers, object keys, and metadata
• HTTP / REST / URL based – easily scriptable, many language choices
• Relatively simple interface compared to file storage
• Scalable to very high object counts
• More easily scaled across multiple geographies
• Ideal for relatively static data
### Block Storage
- Generally SCSI protocol based, organized by LUNs
- Boot volumes for VMs
- Ephemeral vs. Persistent
- Not directly consumed by applications, usually used to hold a filesystem
- Low level storage abstraction upon which file and object storage is built

### File Storage
- Files organized in directory hierarchy and accessed by pathname
- File-based NAS protocols like NFS and CIFS
- Rich and complex application support: random access, in-place file updates, locking, etc.

### Object Storage
- Efficient flat namespace: objects organized by accounts, containers, object keys, and metadata
- HTTP / REST / URL based – easily scriptable, many language choices
- Relatively simple interface compared to file storage
- Scalable to very high object counts
- More easily scaled across multiple geographies
- Ideal for relatively static data

**Cinder**

**Manila**

**Swift**
Cloud Storage 101

Block Storage
- Generally SCSI protocol based, organized by LUNs
- Boot volumes for VMs
- Ephemeral vs. Persistent
- Not directly consumed by applications, usually used to hold a filesystem
- Low level storage abstraction upon which file and object storage is built

File Storage
- Files organized in directory hierarchy and accessed by pathname
- File-based NAS protocols like NFS and CIFS
- Rich and complex application support: random access, in-place file updates, locking, etc.

Object Storage
- Efficient flat namespace: objects organized by accounts, containers, object keys, and metadata
- HTTP / REST / URL based – easily scriptable, many language choices
- Relatively simple interface compared to file storage
- Scalable to very high object counts
- More easily scaled across multiple geographies
- Ideal for relatively static data
OpenStack Block Storage

- **Nova Volume**
  - Originally OpenStack Compute (Nova) included support for ephemeral volumes
    - Used for boot/runtime storage of VMs
    - Ephemeral – volumes only existed as long as VMs
    - Volumes were typically backed by VM server files
  - Nova Volume had limited support persistent volumes on iSCSI
- **Beginning with the Folsom release, a separate persistent block storage service, Cinder, was created**
  - Cinder is a core part OpenStack project
  - Consists of a plug-in interface for supporting various block storage devices
OpenStack block storage - Cinder

Types of block storage

• Ephemeral storage
  • Boot disk, can be additional storage using LVM
  • Local disk attached to each VM host
  • Disappears when a VM is deleted or if a VM host goes away
  • May be backed up (offline) as a glance image

• Persistent block storage
  • Boot or additional storage
  • Exists in a network accessible device, doesn’t go away if VM disappears
  • Typically a Storage array accessed using iSCSI
  • May also be Fiber Channel, an Ethernet block device, or even a file on a NAS box accessed using LVM
  • Must be provisioned and then attached to a VM
  • Open source or commercial options
Cloud scale – block storage

- Cloud applications are characterized by scale-out
  - Large # of small VMs
    - 100’s to 1000’s VM hosts, Many VMs per host
  - Varying utilization
  - More arrays needed to deal with number of VMs and LUNs
- Cloud LUNs are small
  - Std. Boot Vols. - AWS – 7 to 35 GB, HP Cloud – 30 to 960 GB
- Other issues
  - Flat architecture, high volume creation rate
- Scale issues
  - LUNs/array
  - Host connections/array
  - Provisioning latency
Cinder Core Functionality

- **Volumes**
  - Create, Show, Update, Extend, Delete Volume
  - List Volume Summaries/Details
- **Snapshots**
  - Create, Show, Update, Delete Snapshot
  - List Snapshot Summaries/Details
- **Volume types**
  - List/Show volume types
  - Volume types defined in cinder.conf file, “extra specs” for advanced features
- **Other features**
  - Backup to swift or glance
  - Volume migration
  - QoS – extended specs
Drivers are available for

- Ceph RADOS Block Device (RBD)
- Coraid AoE driver configuration
- Dell EqualLogic volume driver
- EMC SMI-S iSCSI driver
- GlusterFS driver
- HDS iSCSI volume driver
- HP 3Par StoreServ Fibre Channel and iSCSI drivers
- HP StoreVirtual / LeftHand SAN
- Huawei storage driver
- IBM XIV/DS8K volume driver
- IBM GPFS volume driver
- IBM Storwize family and SVC volume driver
- NetApp unified driver
- Nexenta drivers
- NFS driver
- SolidFire
- VMware VMDK driver
- Windows
- XenAPI NFS
- XenAPI Storage Manager volume driver
- Zadara
Cinder volume attachment

- **NFS/shared file system**
  - Similar to Nova ephemeral volumes
  - Volume file created on file share, attached through libvirt or other VM specific mechanism
  - All VM hosts must already be attached to file share

- **iSCSI**
  - Attach to iSCSI volume over TCP/IP network
  - Cinder provisions volumes on target, coordinates initiator and target connection

- **FC**
  - Pre-zoned/flat – Grizzly-Havana
  - Zoning - Icehouse
Cinder Futures - Juno

- New drivers – arrays and SAN switches
- Shared volumes – HA
- Bare metal boot
- Volume import/export
- Deeper QoS support
## Cloud Storage 101

### Block Storage
- Generally SCSI protocol based, organized by LUNs
- Boot volumes for VMs
- Ephemeral vs. Persistent
- Not directly consumed by applications, usually used to hold a filesystem
- Low level storage abstraction upon which file and object storage is built

### File Storage
- Files organized in directory hierarchy and accessed by pathname
- File-based NAS protocols like NFS and CIFS
- Rich and complex application support: random access, in-place file updates, locking, etc.

### Object Storage
- Efficient flat namespace: objects organized by accounts, containers, object keys, and metadata
- HTTP / REST / URL based – easily scriptable, many language choices
- Relatively simple interface compared to file storage
- Scalable to very high object counts
- More easily scaled across multiple geographies
- Ideal for relatively static data
OpenStack Object Storage - Swift

- Swift was one of the original components of OpenStack
  - Originally developed by RackSpace
  - Distributed scale-out shared-nothing object storage
- Web APIs for data access
  - HTTP Put, Get, Post, Delete, …
- Swift is open source code, not an API standard
  - There is no compatibility suite and the API is subject to change
  - Enhanced direct implementations
    - SwiftStack, HP (StoreAll), etc.
  - Some vendors implement Swift compatible APIs
    - Ceph, Cleversafe, etc.
  - Some vendors have swift plugins – diskfile layer
    - Gluster, Seagate Kinetic, etc.

Image source: http://docs.openstack.org
Swift Storage Model

- Users authenticate with Keystone
- Each account owns a set of containers
- Containers hold a set of objects
  - Containers have metadata
  - Access permissions at container
- Objects contain data
  - Size is limited, composite objects supported for larger objects
  - Objects also have metadata
  - User specifies object name
  - Name can contain pseudo-paths

Image source: http://docs.openstack.org
Swift Operation

1. Users and applications request to store and retrieve data through the native OpenStack API or the Amazon S3 compatible API.

2. Each data object is stored redundantly by making multiple copies to different nodes (hardware devices). There are no size limits on the objects stored.

3. Just by adding more nodes, clusters are massively scalable to multi-petabyte size and billions of objects.

Image source: http://rogerluethy.wordpress.com
Swift Details

- **The Ring**
  - hash map between names and physical location
  - separate rings for accounts, containers, and objects
  - uses zones, devices, partitions, and replicas.
  - Partitions are replicated, by default, 3 times

- **Proxy Server**
  - responsible for tying together the Object Storage architecture.
  - for each request, it looks up the location and routes the request.
  - public developer API is exposed through Proxy Server.

- **Container Server**
  - primary job is to handle listings of objects
  - doesn’t know where those object’s are
  - listings are stored as SQLite files, replicated across cluster
Swift Details (cont.)

- **Account Server**
  - similar to the Container Server, responsible for list of containers
  - list are stored as replicated sqlite database files
- **Object Server**
  - simple server that can store, retrieve and delete objects
  - objects are stored as binary files on the filesystem
  - metadata stored in extended attributes
  - DiskFile API can be used to access external stores
Swift Details (cont.)

• Replication
  – keeps the system in a consistent state through temporary errors
  – compares local data with remote copies to maintain the latest version.
  – replication updates are push based, updating is just rsync
  – pushes records over HTTP or rsyncs database files.

• Updaters
  – sometimes container or account can not be immediately updated.
  – if an update fails, it is queued locally, the updater will process it
  – eventual consistency

• Auditors
  – Auditors crawl the local server checking integrity
  – If corruption found, file is quarantined, replication will replace
  – other errors are logged
Other Features

- Global clusters
  - Support for geo-dispersed zones – region tier
  - Differing replica counts per region
  - Read from closest replica – based on timing
  - Write affinity

- Bulk requests
  - Auto extract of archive files (e.g., tar)
  - Bulk delete

- Quotas
  - Allows admin to set per account limits (bytes)
Swift Futures - Policies

- Policies
  - Icehouse+
  - Allows user to specify how they want data stored
    - Hardware, encoding, # copies
  - Precursor for erasure codes or other differentiated storage

Image source: http://blog.swiftstack.com
Swift Futures – Erasure Codes

- Driven by Intel, SwiftStack, Box and EVault
- Shards treated like objects, breakup at proxy
- Still in progress, probably Juno+

Put:  
Data:  
Parity:  

Fragments, schematically:
# Cloud Storage 101

## Block Storage
- Generally SCSI protocol based, organized by LUNs
- Boot volumes for VMs
- Ephemeral vs. Persistent
- Not directly consumed by applications, usually used to hold a filesystem
- Low level storage abstraction upon which file and object storage is built

## File Storage
- Files organized in directory hierarchy and accessed by pathname
- File-based NAS protocols like NFS and CIFS
- Rich and complex application support: random access, in-place file updates, locking, etc.

## Object Storage
- Efficient flat namespace: objects organized by accounts, containers, object keys, and metadata
- HTTP / REST / URL based – easily scriptable, many language choices
- Relatively simple interface compared to file storage
- Scalable to very high object counts
- More easily scaled across multiple geographies
- Ideal for relatively static data
OpenStack Shared File Storage

- Not originally planned for OpenStack,
  - Clouds like AWS don’t have shared file storage
  - Security and networking issues are difficult
- Demand has continued for this type of a service
  - For legacy applications that are not object storage enabled
  - Other options like running NFS on a VM or Gluster across nodes are complex to get right
- Originally proposed as an extension to Cinder
  - Proposal from NetApp
  - Implemented in Grizzly Cinder

Image source: http://wiki.openstack.org/wiki/Manila
Manila

- Manila emerged as a separate project over Summer 2013
  - Plan is to make an incubator project in Juno
  - Full project in “K”
  - Still in development, and looking for more participation

- Manila core functionality
  - Create or delete a share
  - Show or list shares
  - Allow/deny access to a share
  - Create/delete share snapshots, list snapshots

- Goal is to provide a file system that is shared between VMs
  - Multiple potential implementations
  - NFS/CIFS direct – Neutron
  - VM mapped
  - Proxy Gateway
Summary

- **OpenStack** is free open source software for implementing public and private clouds
  - Many companies and individuals are involved in development
  - Products and services based on OpenStack exist today, and more are appearing as it matures
- **OpenStack** has support for block, object, and file storage
  - Both open source and commercial storage products supported
  - These provide basic and advanced functionality
  - More features, drivers, implementations are in the works
- **OpenStack** is maturing quickly
  - Give it a try – multiple free and commercial source distributions
  - There is plenty more work to do
  - The development process is open to anyone who wants to help
Questions?

More Information

- http://openstack.org
- http://docs.openstack.org
- https://launchpad.net/cinder
- https://launchpad.net/swift
- https://launchpad.net/manila

Contact me:

- fineberg@hp.com