PURESTORAGE

OpenStack Liberty: A Look at the Glance Image-Cache for Cinder

Simon Dodsley, OpenStack Solutions Architect

27 October 2015

Contents

Summary

OpenStack deployments are rapidly gaining a foothold in many organizations across numerous industry sectors, but the one thing that affects all deployments, no matter who you are using for your deployment tool, whether it be Red Hat, HP Helion or Mirantis, is the time it takes to create a compute instance when using a source image file.

Some third party storage vendors have, in the past, implemented a proprietary method of creating an internal cache of recently used images and using their arrays image cloning technology. With the release of the Liberty version of OpenStack comes the implementation of the Generic Image Cache for Cinder. This creates a unified approach to perform a similar function which can, for backend storage arrays with very efficient cloning method, significantly reduce compute instance creation times. This functionality is a core feature and can be utilized by any backend block storage array.

This paper will discuss this feature and how it has been integrated with the Pure Storage Cinder driver giving implementation details and examples of the volume creation improvements that can be gained.

Standard Approach

Firstly, let's consider the standard methodology for instance creation from images.

When creating an instance from an image that will boot from a persistent storage backend there are at least four OpenStack projects that need to work together. These are

- **Cinder** – the block storage service
- **Glance** – the image service
- **Neutron** – the network service
- **Nova** – the compute service

Let's take relatively simple OpenStack infrastructure with one cloud controller node with the Cinder block storage service configured as well and one Nova compute node. These nodes are configured to communicate through a 1Gb management network and also have 10Gb iSCSI storage and public networks.

When the cloud controller is requested to create a Nova instance that will boot from a Cinder block storage volume using an image that resides within the Glance repository the process that is followed is:

- Create block storage device on block storage backend
- Find the glance image in the repository on the cloud controller
- Attach the block storage volume to the cinder volume node (in this case the cloud controller)
- Copy the image using http over the 1Gb management network to temporary file on cinder volume node

- Write the image to the cinder block storage device utilizing *qemu-img*, allowing the handling of compressed images
- Detach the volume from the cinder volume node
- Attach the volume to the appropriate compute node
- Boot compute instance from block storage device

If the image that is being used is large there is a finite time that can be tens of seconds, which must pass for the image to be copied from the Glance repository to the storage node and then the time to copy the image to the block storage device.

Every time you create a new instance using the same Glance image, exactly the same process has to be followed. You can imagine therefore how long it might take to boot 15, or 50, instances from the same glance image.

Glance Image-Cache Approach

This core functionality that has been added to Cinder in the Liberty release of OpenStack aims to help reduce the time delays encountered when creating multiple instances from the same Glance image.

The first time an image is requested to be used to create a volume on a block storage backend array the same process steps must be taking as detailed above. However, it is at this point that the image cache comes into its own.

The backend array has the option to take a *gold* copy of the image file that has been built on the Cinder backed array. This *gold* image is kept on the array and is owned by an internal Cinder OpenStack tenant. This tenant can be managed like any other tenant including tools like volume quotas and also protects normal tenants from having to see the cached-images, but does not make them globally hidden.

The following screenshots show the creation of a single volume from an image on a Cinder backend and how this is represented from both the Project view of volumes and from the Admin view of the volumes:

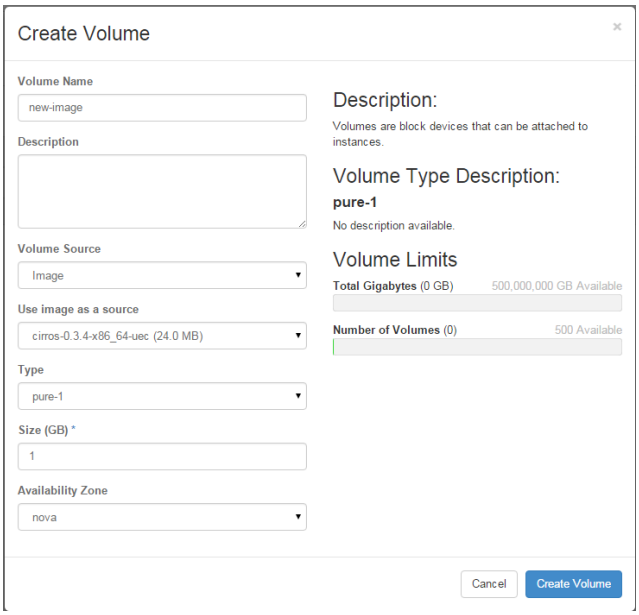


Figure 1: Create a volume in Horizon

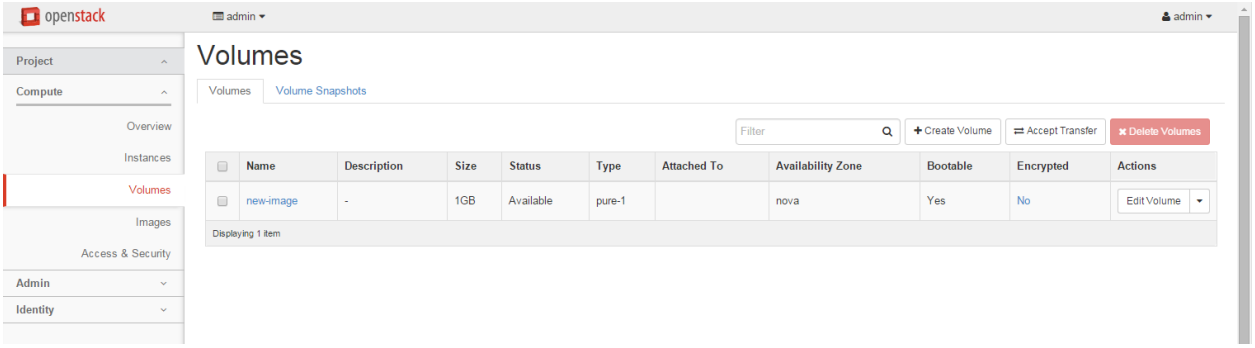


Figure 2: User view of volumes

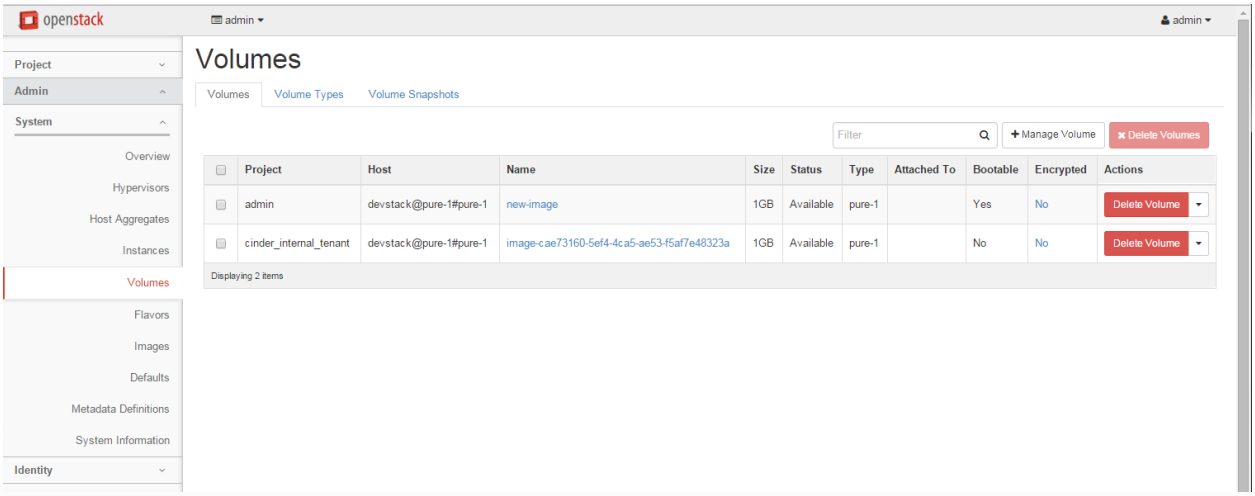


Figure 3: Admin view of volumes

Notice how in the Admin view there are now two volumes. One is the *gold* image-cache copy of the Glance image and the other is the volume that is visible the user who requested the volume initially.

The next time the image is requested from Glance, the glance service will check to see if there is a corresponding image in the volume cache. If the image does exist then OpenStack will use the image clone to create a new volume for the instance. This has now removed the need to copy the image from the Glance node to the storage node with all the associated time savings.

It should be noted that the Image Cache is optional and needs to be enabled in the Cinder configuration file. Details of this are covered later in this paper.

Pure Storage Integration

Using a Pure Storage array as the backed Cinder block storage device will allow the OpenStack Cinder clone image call used by the generic image cache to fully leverage the instantaneous image copy functionality within the Pure Storage FlashArray. The cloned volume is instantly available and can be used by the Nova compute instance to boot. Of course you also gain the performance advantage of booting from an all-flash array with very high read performance and sub-millisecond read latency.

Within the GUI of the Pure Storage array you can see that each volume that is being used by an instance is cloned from the single *gold* copy of the image volume, and that the *gold* volume is not mounted by any host.

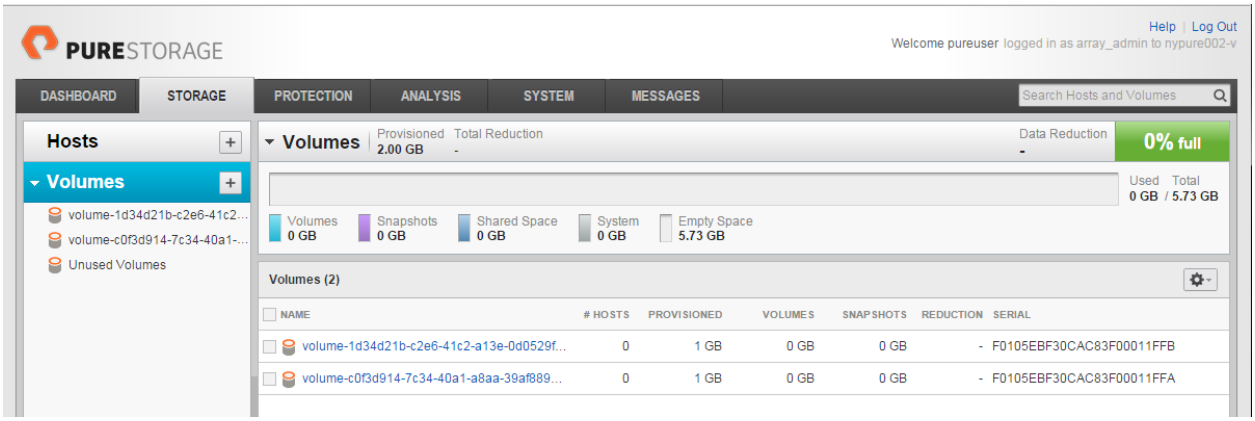


Figure 4: Pure Storage GUI view of created volumes

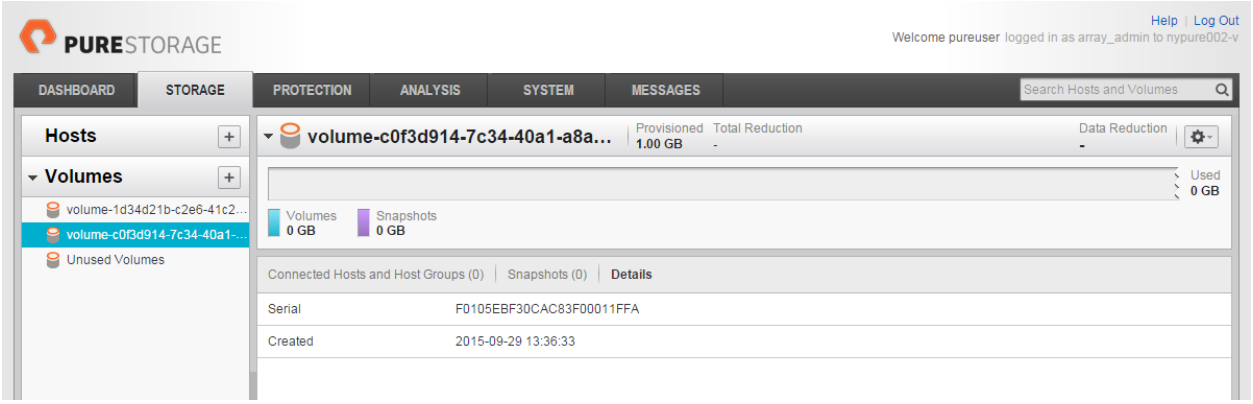


Figure 5: Pure Storage GUI of cached volume

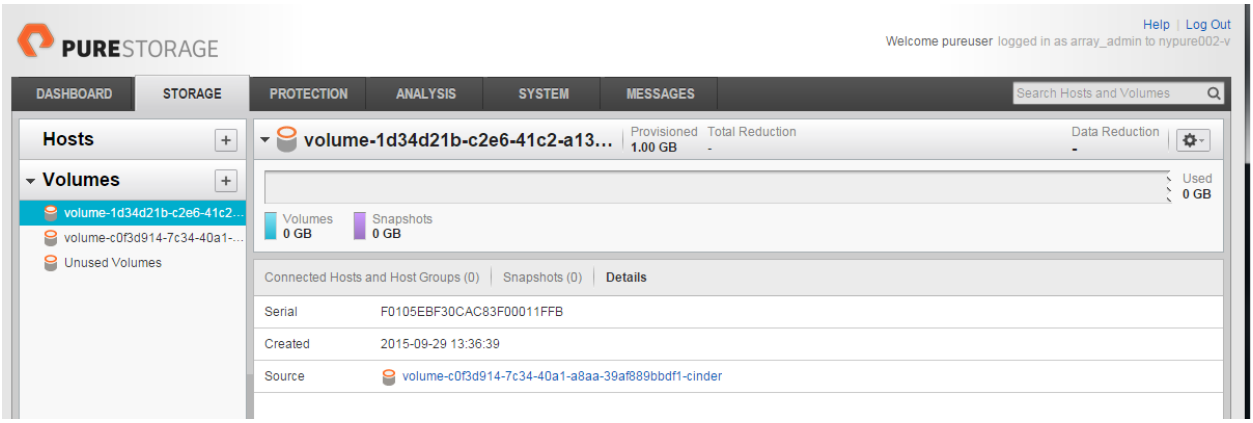


Figure 6: Pure Storage GUI view of cloned volume

Limits and Image Life Cycle

Things are constantly changing in Operating Systems and the same is true of images that are used in cloud implementations within the glance image repository.

It is conceivable that an image that resides in the image cache could be updated in the primary glance image repository, which retaining the same image name. To ensure that this does not cause problems, whenever an image is requested from glance the primary image is compared with the image cache version. Should the image cache version not match exactly with the primary, the cached version is deleted and the primary image has to go through the copy process to the storage node and a new cached copy created.

In OpenStack environments there could be many different images in a glance repository and therefore there has to be limits applied to the capacity of the image cache. This has been implemented using a number of different criteria, configurable in the Cinder configuration file. The criteria that can be used are

- Cache maximum size (GB)
- Maximum number of images

These values are implemented using a logical AND gate. Examples of this are given in the Image Cache Configuration section below.

Cache Management

There is no real requirement to actively manage the image cache as this will be controlled by the core OpenStack code.

If there is a need to delete volumes from the image cache then this can be performed by just deleting the volumes from the OpenStack dashboard as either an admin user or my logging in as the Cinder Internal Tenant user. Deletions can also be performed using standard OpenStack CLI commands.

Configuring the Internal Tenant

As previously mentioned the image-cache uses an internal tenant and project to be configured for the Block Storage service. This tenant owns the cached image volumes but must be configured manually for use and then added to the DEFAULT stanza of the cinder configuration file.

The following commands show how to create and apply these internal tenant and project.

```
# openstack project create —domain default —description "Cinder Internal
Tenant Project" cinder_internal_project
# openstack user create —domain default —password-prompt
cinder_internal_project
# openstack role add —project cinder_internal_project  —user
cinder_internal_project _member_
```

Once the IDs for these are created they should be added into the `cinder.conf` file:

```
[DEFAULT]
...
cinder_internal_tenant_project_id = PROJECT_ID
cinder_internal_tenant_user_id = USER_ID
```

The actual user and project created do not require any special privileges. They can be the Block Storage service tenant or can be any normal project and user.

Configuring the Image-Cache

To enable the image cache the following parameter must be set in `cinder.conf`

```
image_volume_cache_enabled = True
```

This can be applied within either the default stanza of the file, or can be scoped per backend in a specific volume backend stanza when using multi-backends.

As mentioned above there are optional parameters that can be used to control the size of the image cache. These again can be scoped per backed stanza or in the default stanza of `cinder.conf`

```
image_volume_cache_max_size_gb = SIZE_GB
image_volume_cache_max_count = MAX_COUNT
```

The default values for these are 0, which means unlimited.

For example, a configuration which would limit the maximum size to 200GB and 50 cache entries would be:

```
image_volume_cache_max_size_gb = 200
image_volume_cache_max_count = 50
```

If you have an implementation of OpenStack with many images that are being put into the cache when the count limit is reached, or the capacity limit of the cache is reached, then the least recently used image will be evicted from the cache to create space for the new image.

Ceilometer Integration

As part of the implementation of the generic image cache there are three cache actions will trigger telemetry messages to Ceilometer.

These messages are:

- `image_volume_cache.miss` – A volume is being created from an image which was not found in the cache. This will usual mean that a new cache entry will be created for the image
- `image_volume_cache.hit` – A volume is being created from an image which was found in the cache
- `image_volume_cache.evict` – A cached image volume has been deleted from the cache

Performance Improvements

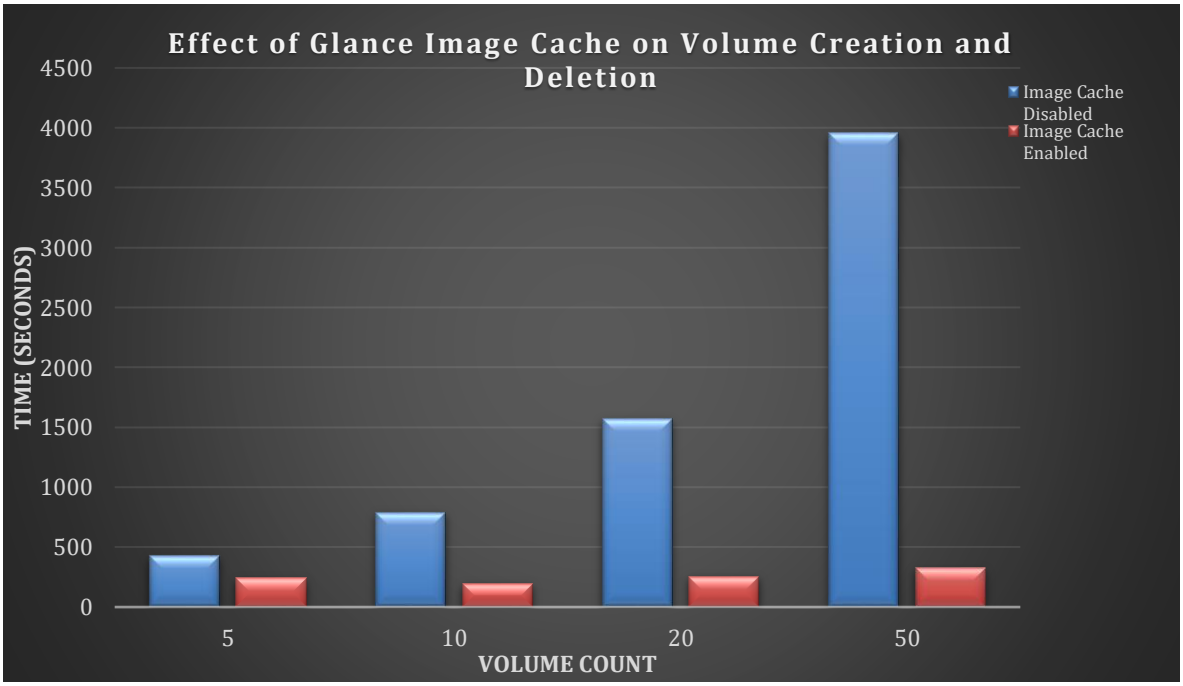
To illustrate the performance gains achieved using the image cache the following results come from a series of tests we did using a Rally Benchmark Test to automate the volume creation and deletion tasks.

The block storage backend was a Pure Storage FlashArray and the Glance repository was populated with a single 20GB CentOS raw image.

The following table and graph show times to complete creating and then deleting multiple volumes with the image cache turned on and off and allowing for 2 processes to occur in parallel.

Number of Volumes (Creation and Deletion)	Time Elapsed (seconds)	
	Image Cache Disabled	Image Cache Enabled
5	426	241
10	790	197
20	1,573	257
50	3,952	330

Table 1: Time to Create and Delete Multiple Volumes



Remember that these times are for both the creation and deletion of volumes. The detailed results showed that the time for volume deletion was unaffected by the image cache, as would be expected and on average took 2.25s per volume.

If we remove the image deletion times from the above results we get the following, again with a parallelism of 2:

Number of Volumes (Creation Only)	Time Elapsed (seconds)	
	Image Cache Disabled	Image Cache Enabled
5	373	232
10	773	180
20	1,545	136
50	3,781	264

Table 2: Time to Create Multiple Volumes

To simplify these numbers even more we can look at the non-parallelized average volume creation times we get the following:

- Image Cache Disabled: 154s
- Image Cache Enabled: 2.9s (*first volume creation excluded*)

The difference in volume creation time is significantly better with the Glance Image-Cache for Cinder enabled and obviously this will have a major impact in time to service when creating instances for users.

Summary

It is clear that the Glance Image Cache feature can have a significant effect on the volume creation times when using a standard image file.

In an environment where OpenStack instances are being created and deleted on a large scale this could significantly increase productivity rates and reduce wait times for instances to be made available to a user base.

An added benefit to the use of the glance image cache it that your internal networks do not have to deal with the additional load of the multiple http-based copies that historically had to be dealt with for each volume creation.

About the Author

As Global Solutions Architect, Simon is helping implement OpenStack technologies on Pure Storage. Core items include best practices, reference architectures and configuration guides.

With over 25 years of storage experience across all aspects of the discipline, from administration to architectural design, Simon has worked with all major storage vendors' technologies and organizations, large and small, across Europe and the USA as both customer and service provider. He also specializes in Data Migration methodologies assisting customers in their Pure Storage transition.

Blog: http://www.purestorage.com/blog/author/simon

PURESTORAGE

Pure Storage, Inc.
Twitter: @purestorage
www.purestorage.com

650 Castro Street, Suite #260
Mountain View, CA 94041

T: 650-290-6088
F: 650-625-9667

Sales: sales@purestorage.com
Support: support@purestorage.com
Media: pr@purestorage.com
General: info@purestorage.com