Implementation of Storage Tiering Using OpenStack to Improve Data dynamics on Cloud Computing

Bhagyashri Kulkarni¹, Varsha Bhosale²

¹PG Sholar, Mumbai University, Department of Information Technology, Vidyalankar Institute of Technology, Wadala(E), Mumbai, India.
²Associate Professor, Mumbai University, Department of Information Technology, Vidyalankar Institute of Technology, Wadala(E), Mumbai, India.

Abstract
Tiered storage is the assignment of different categories of data to different types of storage media in order to reduce total storage cost.

Tier 1 data is frequently accessed critical data and it should be stored on reliable and high-quality media. Tier 2 data is seldom-used less critical data and should be stored on less reliable media. As the tier number increased, less reliable (hence cheaper) media could be used. Thus, tier 3 in a 3-tier system might contain rarely used data.

In OpenStack, Cinder component provides Block storage capabilities. Block storage volumes created by Cinder provide persistent block level storage for compute instance (Virtual machines). Cinder has support for numerous storage platforms. Cinder also provides support to have different type of storage devices in the backend. Storage tiering solution for OpenStack moves data across different Cinder backend based on frequency of use to save overall storage cost.

Keywords: Storage, OpenStack, Cinder, Security.

1. INTRODUCTION

[11] With every new cloud service consumed the total amount of content that needs to be stored and accessed is growing substantially. However, most cloud consumers’ care about is how fast the storage content is accessed and consumed rather than where or on which type of disk it is stored on, leading to increase in the cost of the cloud storage and the efficiency of the cloud based services. Storage collectivism in cloud does not work since you can’t treat all cloud content in the same way and a tiered storage for cloud services is a must for clients to counterbalance cost with performance. A tiered storage strategy, properly planned and executed, can increase service levels for critical applications, while reducing the overall cost of data storage.

Storage Tiering involves

• Grouping data into different categories
• Assigning categorized Data to different types of storage media.

Storage Media selection is based on performance, protection and recovery speed.

Logical grouping as per usage: Tier 1, Tier 2 and Tier 3 Storage:

• Tier 1 data (mission-critical, recently accessed files) might be stored on expensive and high-quality media
• Tier 2 data (financial, occasionally -used) might be stored on less expensive media in conventional storage
• Tier 3 Data (event-driven, rarely used, or unclassified files) on recordable tapes.

2. METHODOLOGY

OpenStack is more than a single piece of software: It is a specification that defines a set of functions required to implement cloud solutions [6]. Components that implement these functions communicate via well-defined interfaces. This is more than reference architecture components of this stack are implemented as various subordinate elements or services, each known by its project code name [6]:

• Keystone – Identity service
• Horizon – Web GUI
• Nova – Compute service
• Glance – Image service
• Neutron – Network services (formerly called Quantum)
• Cinder – Block storage service
• Swift – Object storage service

Here are some facts that we know about OpenStack Cinder component.

• Cinder component provides Block storage capabilities in OpenStack.
• OpenStack Block Storage service is not a shared storage solution. (i.e. one volume cannot be attached to multiple compute instances)
• OpenStack Block Storage service provides support to several vendors’ back-end storage devices.
• Vendor specific block storage driver can perform storage tiering within devices supported by that driver.
• Havana release of OpenStack has introduced the ability to migrate volumes between back-ends. (very limited support by vendors)
• Vendor neutral solution is not available in the current releases.

Below mentioned options were considered to decide the level at which Data tiering should happen.

**Block level**

In this approach, we need to monitor the activities happening at the block level to decide which particular blocks are suitable for moving to next tier of storage. Data movement also has to be handled explicitly. Though very good storage utilization can be achieved, it would be very complex to monitor, move and reconstruct the data when needed. Performance of the solution would be poor because of very complex logic and large data movements. Suppose due to some reason, we want to cancel tiering operation and roll back to the state prior to tiering, implementing such roll back mechanism will also be complex in block level.

**File level**

In this approach, file modification timestamp can help us deciding suitable files for the movement. OpenStack does not provide any mechanism for file movement so complex logic needs to be written for file movement. Though better storage utilization can be achieved, it would be complex to move files around and reconstruct the data when needed. Performance of the solution would be poor because of complex logic and large file movements. Implementing roll back mechanism will be easy than block level.

**Volume level**

In this approach, we need to monitor the operations (read/write) happening at volume level and then based on tiering policy move the complete volume at cheaper storage. OpenStack supports APIs for volume snapshot creation and volume creation from the snapshot. Telemetry services provide us mechanism for volume monitoring that can help us deciding suitable volumes for the movement. Though storage utilization might not be the best, it would be easy to build the solution at this level. As OpenStack provided functionality can be used for tiering operation, we can design a solution that is vendor neutral. Implementing roll back mechanism would be easy as OpenStack supported operations are used.

**Comparison of Storage tiering options**

<table>
<thead>
<tr>
<th>Features</th>
<th>Comparison of Storage Tiering Option</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Block Level</td>
</tr>
<tr>
<td>Storage Utilization</td>
<td>++++</td>
</tr>
<tr>
<td>Complexity</td>
<td>++</td>
</tr>
<tr>
<td>Performance</td>
<td>++</td>
</tr>
<tr>
<td>Rollback</td>
<td>++</td>
</tr>
</tbody>
</table>

Based on the above analysis, we decided to go for Volume level Storage Tiering.

There could be multiple factors based on which we can decide the tiering policy like data criticality, performance requirement, protection needed, frequency of use etc. We are planning to base our solution on ‘frequency of use’ factor.

3. **SOLUTION**

As discussed earlier, in OpenStack, Cinder component provides block storage capabilities for compute instance and it supports numerous storage platforms in the backend. We can use different type of storage devices at the backend of Cinder component and provide storage tiering capability in OpenStack by moving less used data to cheaper storage. In the diagram above we can see 4 types of storage devices attached in the backend and data movement between them based on frequency of usage.

If there is no activity performed on a volume for the pre decided period, we can take a snapshot of that volume and create a volume on next tier storage device using that snapshot. Once the operation is successful, we can make changes so that running VM instance will point to this newly created volume and original volume is deleted freeing the space on expensive storage. Similar logic can be implemented for multiple tiers.

We just need to make sure that we rollback to original volume and cancel the tiering if during the tiering operation volume data gets changed. For unsuccessful operation, we will delete the newly created volume and snapshot created for tiering purpose.

Reverse logic can also be implemented, if customer tries to access volume at the secondary storage. Customer will continue accessing the volume though at some slower speed. Once the window for the reverse operation is available we will move the volume to the primary storage.

At a high level, to solve the problem, we need to perform two major tasks mentioned below:

1. Monitor volume activities so that we can decide upon frequently used and rarely used volumes.
2. Migrate volume to cheaper or expensive media based on tiering policy.

Volume Monitoring
Volumes need to be monitored for activities to decide upon frequently used and rarely used volumes. Based upon usage data and pre-defined tiering policy volumes to migrate can be identified.

Telemetry service for Nova measurement can be used to monitor instance activities. We can use meters defined for compute instance in ceilometers to monitor instance activities. disk.read.requests (Number of read requests) and disk.write.requests (Number of write requests) meters gives us cumulative counts of read and write activities. We can monitor these values periodically to see the instance usage pattern.

We can use nova diagnostic commands to retrieve volume usage information. It gives read_request and write_request values for all the volumes attached with VM. We can monitor these values periodically to find out if the volume usage and the tiering policy can then be applied.

Volume Migration
Once the volumes are identified, their migration to cheaper or expensive media based on tiering policy can be done.

Following are the possible ways of volume migration.

1. Using ‘cinder migrate’ operation
   Migration of volume that has snapshot created is currently not supported. Following are the possible workflow scenarios for Volume migration.
   a) Volume is attached to a VM instance
      If the volume is attached to an instance, Block storage service can create a new volume and can call Compute service to copy the data from original to new volume. Currently this is supported only by the Compute libvirt driver.
   b) Volume is not attached
      If the volume is not attached to any instance, Block storage service can create a new volume and copies the data from original to new volume. While most back-ends support this function, not all do.

2. Creating snapshot of volume to be migrated and then creating volume again from the snapshot at desired location.
   When direct support to volume migration is not available, we can create a snapshot of a volume and then create a volume at cheaper storage using that snapshot.
   • Create snapshot of the volume to be migrated (“cinder snapshot-create …” command)
      The snapshot will get created in Object Storage. (Swift component of OpenStack)
   • Create volume from that snapshot at the desired location (“cinder create --snapshot id …”)
   • Change the VM instance to point this newly created volume.

Option 2 serves us better as it is not dependant on ‘cinder migrate’ support and also the limitation of migration not allowed when volume has snapshot is not there.

Storage tiering will be performed on a volume when it is in one of the 2 states, AVAILABLE and IN-USE. Tiering operation will not be performed when the volume is in any other states.

Technology Stack:
2014, ANALYST(S): LYDIA LEONG, DOUGLAS TOOMBS, BOB GILL, GREGOR PETRI, TINY HAYNES: G00261698

AUTHOR

Ms. Bhagyashri N Kulkarni is a PG Scholar in Information Technology Department at Vidyalankar Institute of Technology, Mumbai. My research areas are Cloud Computing, Cloud Infrastructure Management and Mobile Communication and Computing.

Ms. Varsha Bhosale is working as Associate Professor in Information Technology Department and also Wise Principal at Vidyalankar Institute of Technology, Mumbai. She has 15 years of teaching experience. Her research areas are Image Processing and Data Mining.