CoprHD: Bypassing the virtual layer of storage abstraction for advanced users.

Dr. Udaya Rani V, Senior Associate Professor,
Data Engineering and Cloud Computing,
School of Computing & IT, REVA University,
Bengaluru
udayamurthy@revainstitution.org

Maheshchandra. C. S,
IV semester M.tech.,
School of Computing & IT, REVA University,
Bengaluru.
cs.maheshchandra@gmail.com

***Abstract*- A typical data center environment has multi-vendor, heterogeneous and all types of storage systems (bock-based, file-based, object-based). Software-defined storage (SDS) is a storage infrastructure used to maintain and manage the data center by abstracting the physical storage systems. CoprHD is an open-source SDS product from DELL-EMC, designed and developed primarily to create and manage virtual data centers. CoprHD is used extensively by enterprises and storage service providers to manage their data centers and provision the storage for their customers respectively. Primary aim of this project is to bypass the virtual layer of storage abstraction to gain direct access to the physical storage pools which is desirable in certain scenarios.**

Keywords—Virtual Pool, Virtual Array, Software Defined Storage.

I. INTRODUCTION

In a traditional data center, there are several challenges in provisioning and managing storage in an efficient and cost-effective manner. Some key challenges are - each application type normally has its own vertical stack of servers, networking, storage, and security. This leads to the creation of a loose collection of IT silos, which increases the infrastructure's complexity. This creates management overhead and increases operating expenses. It also leads to poor resource utilization because capacity cannot be shared across stacks. Data centers have multi-vendor, heterogeneous storage systems, and each type of storage system (block-based, file-based, and object-based) has its own unique value. However, critical functionality is often tied to specific storage types, and each storage system commonly has its own monitoring and management tools. There is limited resource sharing, no centralized management, a little automation, and a lack of standards in this environment. Application workload complexities and higher SLA demands pose a further challenge to IT. IT finds it difficult to allocate storage to satisfy the capacity requirements of applications in real time. There are also new requirements and expectations for continuous access and delivery of resources most likely as in a cloud environment. Traditional environments are not architected for third platform technologies such as cloud computing, Big Data analytics, and mobile applications. Therefore, there are several challenges in managing massive data growth, cost-effective scaling, and providing self-service access to storage. These challenges have led to the advent of the software-defined storage model.

II. SOFTWARE DEFINED STORAGE

Software-defined storage (SDS) is a storage infrastructure that is managed and automated by software. Fig. 1, depicts the generic SDS's architecture. SDS abstracts heterogeneous storage systems and their underlying capabilities, and pools the storage resources. Storage capacity is dynamically and automatically allocated from the storage pools based on policies to match the needs of applications [4].

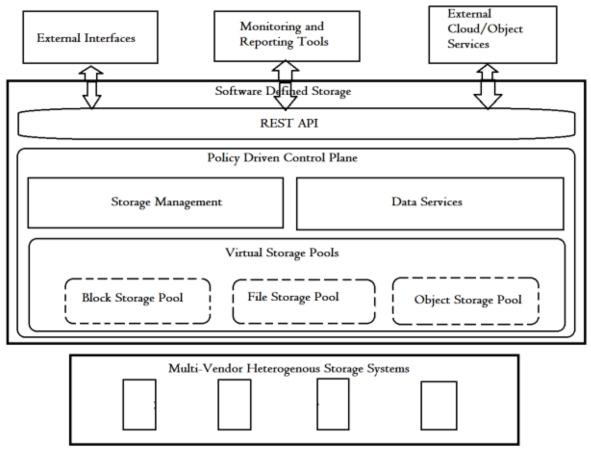

Fig. 1, Software Defined Storage Architecture

In general, SDS software abstracts the physical details of storage (media, formats, location, low-level hardware configuration), and delivers storage as software. A storage system is a combination of hardware and software. The software stack exposes the data access method such as block, file, or object, and uses persistent media such as HDD or SSD to store the data. SDS software separates the software layer of a storage system from the hardware. It supports combinations of multiple storage types and access methods, such as block, file, and object. It enables storing data on both storage systems and commodity disks, while providing a unified external view of storage. This allows organizations to reuse existing storage assets, and mix & match them with commodity resources, while serving data through a single name-space and storage system spread across these different assets. For example: In a data center that contains several distinct file servers, SDS can provide a global file system, spanning the file servers and allowing location-independent file access. This is similar to how the local file system on a compute system hides the underlying disk block structure and manifests file access interface to the storage media.

SDS enables organizations to build modern, hyper-scale storage infrastructure in a cost-effective manner using standardized, commercial off-the-shelf components. The components individually provide lower performance. However, at sufficient scale and with the use of SDS software, the pool of components provides greater capacity and performance characteristics.

III. COPRHD

CoprHD is an open source version of ViPR [4] software platform for building a storage cloud [2]. Fig. 2, show It is designed with two key goals in mind:

1. Make a data center full of storage, compute & network resources (block arrays, filers, SAN and NAS switches) look and feel like a single massive virtual storage array, automating and hiding its internal structure and complexity.

2. Extend to the end-users, a full set of essential provisioning operations. It supports operations as simple as block volume creation, to fairly complex operations such as creating disaster-recovery-protected volumes in different data centers. A key principle of this capability is to allow them to do it in a truly multi-user, multi-tenant fashion by providing tenant separation, access control, auditing, usage monitoring, and other features.

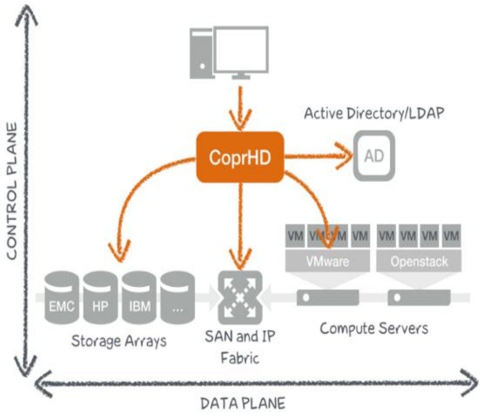

Fig. 2, CoprHD [3]

CoprHD addresses these goals in a vendor-agnostic and extensible way, supporting a variety storage devices from several vendors, without resorting to a 'least common denominator' approach in regard to features offered by these devices. CoprHD holds an inventory of all storage devices in the data center and understands their connectivity thus it itself does not provide storage. It allows the storage administrator to group these resources into either:

1. Virtual pools, with specific performance and data protection characteristics or,
2. Virtual arrays, segmenting the data center infrastructure along lines of fault tolerance, network isolation, or tenant isolation.

A single uniform and intuitive API is used by end users to provision block volumes or file shares on virtual pools or virtual arrays in a fully-automated fashion. CoprHD automates all hidden infrastructural tasks, such as finding optimal placement for a volume or necessary SAN fabric configuration. Also, while exporting volumes, CoprHD adds intelligence of its own with features like the ability to apply metrics-based algorithms to port selection. Finally, CoprHD provisioning goes beyond creating volumes, it has the ability to perform complex orchestrations that cover the end-to-end provisioning processes including volume creation, SAN configuration and mounting the newly created volume on a host. Also, it has built in support for environments that include VMware and Vblock# technologies [3].

IV. BYPASSING THE VIRTUAL LAYER OF STORAGE ABSTRACTION FOR ADVANCED USERS.

CoprHD greatly simplifies discovery and self-service provisioning of storage services. However, due to abstraction, the control on Physical devices is lost. Sometimes it is desirable to have control of physical devices to do pass-through operation.

Abstraction has a lot of advantages, but it comes with a few cons. Two main disadvantages are listed below:

1. Leads to lack of low-level control on storage devices. CoprHD provides and abstract layer of virtual pool (vPool) and virtual array (vArray), and it creates volumes on top of vPools and vArrays. But what if administrators want to create volumes on physical storage pools directly?
2. Lack of uniform access/methods for all underlying storage devices. The User just want to list of devices and operations supported on those devices as it will allow future expansion of CoprHD on currently unsupported device types.

This project basically bypass the virtual level of complexity involved with CoprHD and gives direct access to storage devices leveraging the CoprHD framework. This project is useful for those users who have the complete knowledge about the configuration of physical storage systems in their data center environment, want to have access to low-level physical system configuration, still leveraging the CoprHD platform.

A. Implementation Methodology

In this project, the VPLEX# driver is implemented and the steps for implementation are as described below. Meanwhile, Fig. 3, gives the control flow of the VPELX# bypass virtual layer implementation.

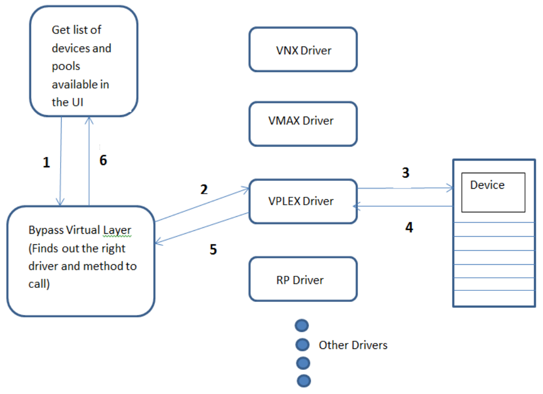
Get list of devices and pools available in the UI
1
6
VNX Driver
VMAX Driver
2
VPLEX Driver
3
4
5
Device
Bypass Virtual Layer (Finds out the right driver and method to call)
RP Driver
Other Drivers

Fig. 3, the control flow for VPLEX# bypass virtual layer.

1. Select the storage system and storage pool on which the volume has to be created. This information will be available in the personalized UI built exclusively for this project. These storage devices and their pools are pre-discovered in CoprHD. The bypass virtual layer driver, which receives the REST API call validates it for its authenticity and further chooses the specific storage driver based on the storage system type.
2. The Storage driver which now has the information of the specific storage system and its associated storage pool creates a workflow to execute the task(create/delete), and further contacts the physical device.
3. At the device, the volumes are created on the specific storage pool specified.
4. The status of the task is updated to the storage driver. Valid statuses are success & failure. In the event of failure the cause for the same is notified and the progress if any is rolled back.
5. The status of the workflow is updated to the bypass virtual layer driver, the state of the workflow is saved.
6. The status of the task is displayed on the UI. If the task succeeded, then it is displayed in green color. Apparently, if it fails it is displayed in red color and the reason for the failure is also displayed.

This project is implemented using Java – for backend, AngularJS – for frontend. Apache Cassandra database is used to persist the data. Apache Zookeeper is used for workflow handling.

V. CONCLUSION

The paper shows the implementation of the driver which bypasses the virtual layer of abstraction present in CoprHD. The beneficiaries of this project would be those who have the knowledge about the storage systems and its associated storage pools present in their data center environment. This project is definitely a value add to those users who need access to low level information of the storage systems while leveraging the CoprHD platform.

REFERENCES

[1] CoprHD Home [Online]. Available : https://coprhd.atlassian.net/wiki/display/COP/CoprHD+Home
[2] CoprHD Architecture [Online]. Available : https://coprhd.atlassian.net/wiki/display/COP/Architecture
[3] A Short Guide to the CoprHD Architecture [Online]. Available : https://coprhd.atlassian.net/wiki/display/COP/A+Short+Guide+to+the+CoprHD+Architecture
[4] Dell EMC ViPR Controller [Online]. Available : https://www.emc.com/collateral/data-sheet/h11750-emc-vipr-software-defined-storage-ds.pdf
[5] CoprHD: ScaleIO Driver based on Southbound SDK [Online]. Available : https://ir.library.oregonstate.edu/xmlui/handle/1957/60008
[6] Intel White Paper EMC ScaleIO and CoprHD Overview [Online]. Available : http://www.intel.com/content/dam/www/public/us/en/documents/white-papers/emc-scalelo-coprhd-overview.pdf