TEST Engineering & Management

May-June 2020
ISSN: 0193-4120 Page No. 5276-5281

Seamless Persistent Storage Availability for Stateful Application Running on Kubernetes Platform

[1]Sai Kumar D, [2]Vishwanath Y,
[1,2]School of Computing & Information Technology, REVA University, Bengaluru, Karnataka, India

Article Info
Volume 83
Page Number: 5276-5281
Publication Issue:
May-June 2020

Article History
Article Received: *19 November 2019*
Revised: *27 January 2020*
Accepted: *24 February 2020*
Publication: *16 May 2020*

Abstract
Containers are rapidly being adopted by all sizes of enterprises which they use them to quickly build, deploy, and scale cloud native applications. Containers, along with containerization technology like Docker and Kubernetes, are used for deploying applications in cloud. Containers are completely isolated environment in which they have their own processes or services like virtual machines except they all share the same OS kernel. Kubernetes is a container orchestration tool which is used to deploy and manage containers. Containers were introduced to package microservices runs as stateless state& managing stateful application is difficult. To solve this issue, where container can run stateful application and store its information we need persistent container storage that is compatible across physical, virtual and cloud infrastructures. The main objective of this project is to leverage Kubernetes platform to create management layer for database service on Ceph Storage using Rook Orchestration. As part of this project ,Operator pattern is used in which management layer uses Kubernetes API for infrastructure & ETCD database for persistent storage

Keywords: *sizes of enterprises, cloud native applications.*

1. Introduction

IT infrastructure has rapidly over past decade. With the ease of cloud computing & advancement of virtualization technology all business-critical workloads are provisioned & managed in cloud infrastructure. Initial days workload was run in 1:1 fashion with each workload used to run on dedicated hardware with underutilizing lot of resources in terms of CPU, Memory, Storage & Network. Once virtualization came into existence it has brought ability to run all applications in single piece of hardware. Many cloud providers started providing on demand compute resources through Infrastructure as a Service (IAAS) through virtualization technology Virtualization used technology of hypervisor which emulates single piece of hardware & creates a layer to run multiple operating on same physical machine. Each virtual machine needs guest Operating System, virtual copy of hardware resources. Due to increased virtual copy, resources and moving of VMs between public & private cloud can be challenging. To overcome this issue containers came to existence which shares kernel of host OS. Containers
light-weight & multiple containers can be deployed on same host OS.

The rise of containerization technology has transformed software development lifecycle & paved way for migrating applications from monolithic to microservices architecture.

Container offers virtualization at the Operating system level using abstractions like chroot, namespace and cgroups,

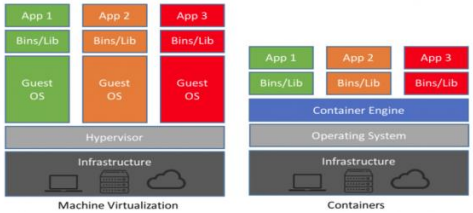

Figure 1: Comparison of Virtualization vs Containers

Published by: The Mattingley Publishing Co., Inc.

May-June 2020
ISSN: 0193-4120 Page No. 5276-5281

required application packaging offers by container engine like Docker and orchestration for containers delivered through Container Orchestrator like Kubernetes. Free and Open-Source (FOSS) adoption helped enterprises to avoid vendor lock-in, de-facto standard software's like Docker (Container Engine) and Kubernetes (Container Orchestrator) emerged as main players and gathered the trust of enterprise to adopt for production needs

With rise of containerized applications starting single instance of containers is easy job but when we have lot of application to run on multiple physical node it is cumbersome process. For this purpose, to schedule container and run them we need container orchestration tool to manage, deploy & scale without any manual intervention. Kubernetes is a container orchestration tool for scheduling pods, manage workloads &verifying health status of pod. Pod is smallest deployable object which is wrapped around one or more containers.

Storage plays one of the key roles in computing. Earlier data was stored in tape drives it got evolved into Hard Disk drives & Solid-State Drives. For enterprise level storing data & preventing it from point of failure is also important. Raid controllers came into existence which help to create virtual drives by implementing fault tolerance & grouping multiple hard disk as single disk virtually. With the cost of storage growing exponentially with advancement of technology open source offers solution of Software Defined Storage Solutions (SDS). In SDS, storage is managed by underlying software for creation, placing of data, maintaining quorum, redundancy.

In monolith architecture application which run are in stateful state where data about each client session is saved and uses it when client makes a request next time. In microservice architecture application runs in stateless state where it does not save client data generated in session for use in next session with that client. In containerized environment workloads are stateless application where it does not store any data once container is terminated. Docker uses Layered architecture model which consists of image layer and container layer. All files inside container are by default stored under container layer. Images are template for building base OS. We use this template to create container. Once container is created it adds writable layer on top of immutable image. All data written on container will get stored in writable container layer till process is running once it gets exited all data will be destroyed. Managing of writing data into container image is performed by storage driver. Storage driver provide union file system which is tightly coupled with Linux Kernel. Adding extra abstraction with directly impact performance of container. To overcome this issue docker has provided two solutions of volume mount and bind mount.

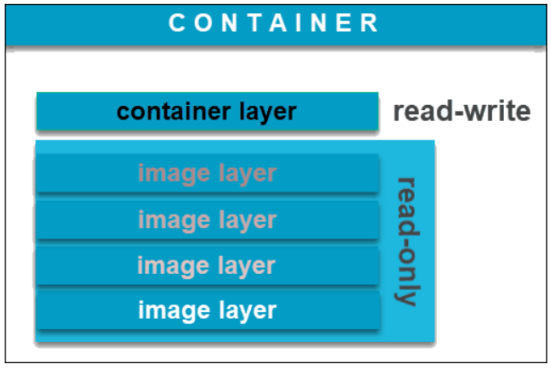

Figure 2: Docker Container Image with layers

There are many container orchestration tools like Docker Swarm, Apache Mesos and Kubernetes. Docker Swarm and Apache Mesos are mostly of enterprise licensed version which involves additional cost whereas Kubernetes is completely open sourced and maintainer by larger community so for this research we will be working with Kubernetes platform. As part of this research it covers running of stateful application with persistent storage on Kubernetes environment. Below are the motivated Research Questions (RQ) that covers the extent of the persistent storage on Kubernetes areas as part of this research paper:

- RQ1: What storage should be used for deploying stateful application?
- RQ2: How to deploy & manage storage in containerized environment? Rook
- RQ3: How to make storage available for Kubernetes cluster?
- RQ4:How does stateful application will store data with storage deployed?

The further sections of this paper organized as follows: Section-II provides the overview of literature survey on RQ's with Research Method Steps (RMS1-4). Section-III Addresses Overview &Integration of persistent storage availability on Kubernetes platform. Section-IV describes the test approach, evaluation and outcomes. Finally, Section V concludes the paper and outlines the future research steps

2. Literature Survey

An extensive literature review carried out on Integrating stateful containerized application on Kubernetes with persistent storage. Total fifteen research papers are identified that are related to this paper RQ's, research work of those papers provided the right direction of what worked and what did not work. Below are points that influence the research of this paper.

RMS1: Kubernetes supports several types of standard storage solutions such as StorNXT, GlusterFS, EMC ScaleIO, Fibre Channel, CephFS, VMWare VSAN, or public cloud solutions like Amazon AWS EBS, Azure Disk or File, Google's Persistent Disk.

Published by: The Mattingley Publishing Co., Inc.

May-June 2020
ISSN: 0193-4120 Page No. 5276-5281

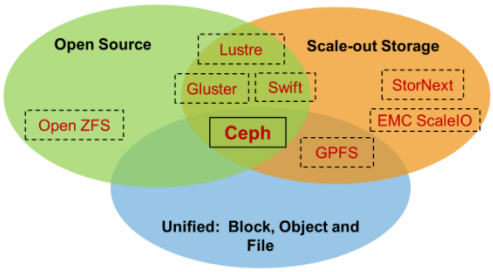

Figure 3: Uniqueness of Ceph storage compared to other solutions

Ceph Storage is unique storage compared to other available storage solution. Key points taken into consideration for considering Ceph Storage

- Open Source Solution
- Software Defined Storage
- Support Enterprise level storage up to Peta-Bye Level
- Supporting all storage types like (Block, Object, File Level)

Most of the above solution have software defined or open source solution but none of them offered all 4 options. Ceph storage includes many enterprise solutions like fault tolerance, redundancy, snapshots, self-healing, provisioning,
Auto-scaling. Ceph storage can be run on commodity hardware without spending much on traditional hardware as there is no vendor lock-in. Ceph is maintained by RedHat Enterprise which has good community support. It supports huge amount of data storage in terms of Petabyte & stores different types of data like Block & Object.
RMS2: For deploying and orchestration of containers we have Kubernetes platform which will take care of all tasks associated with container management. Rook is a cloud-native, open-source storage orchestrator for Kubernetes. Rook couples together Ceph storage and Kubernetes platform together to deliver high performance and automatically scaling storage workloads. Rook is a Kubernetes storage operator which helps to deploy and manage Ceph clusters
RMS3: To integrate Ceph storage to Kubernetes we will use Container Storage Driver(CSI). CSI driver has three module which will use to interact with Ceph Storage i.e. CSI Identity is used for identifying Ceph plugin and returning in healthy drive information to control manager, CSI Controller is used for controlling and managing the volume, CSI Node is used for managing volume's action in node

RMS4:Kubernetes manages volumes and container using metadata. The metadata inform is passed in yaml manifest file as Persistent Volume – the low-level representation of a storage volume, Volume Driver – the code used to communicate with the backend storage provider, Pod – a running container that will consume a Persistent Volume,
Persistent Volume Claim – the binding between a Pod and Persistent Volume, Storage Class – allows for dynamic provisioning of Persistent Volume

3. Kubernetes & Ceph Storage With Rook Architecture

Kubernetes is a container orchestration tool developed by Google. It is an open-source and supported by many enterprise organizations. The main motive of Kubernetes is to run applications in the form of containers in an automated way so that it can be easily deployed, maintained & scaled. Kubernetes follows distributed systems paradigm where a cluster i.e. group of nodes that appears as a single system. Distributed system comes with a concept of master-worker node. In this system more than one node can be used as master for high-availability. Master node plays an important in managing nodes and hosting all necessary services which are used to monitor all worker nodes. It stores the member information regarding the different nodes, planning which containers are scheduled. When a worker node fails it migrates all processes from failed worker node to healthy worker node, Kubernetes master takes care of all activity like scheduling, provisioning, configuring and exposing API's to client. All the activities are done by master node using the component called control plane components.
Four main basic components of master node (control plane)

1. API Server
2. Scheduler
3. Controller Manager
4. ETCD

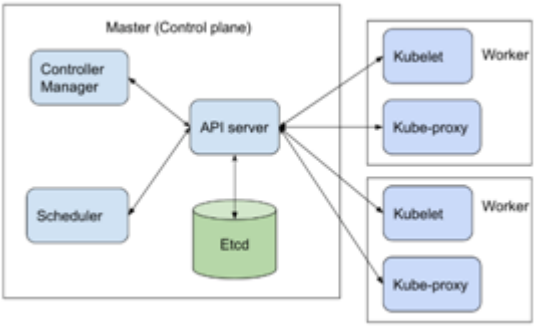

Figure 4: Kubernetes master & worker node architecture

The **API server** is a main component using which all cluster components communicate with each other. API server is gatekeeper for entire cluster. Scheduler, controller manager and another worker node component communicate with the API server& it exposes API to all other component for almost every operation. In Kubernetes user can interact with API server using kubectl utility

Published by: The Mattingley Publishing Co., Inc.

Scheduler is responsible for physically placing pods across multiple worker node. It will automatically detect pod goes which worker node based on the resource requirements, set of constraints defined in configuration file. It will aptly find out the appropriate node which satisfies all requirements to run the pod in worker node

The **Controller Manager** keeps track of whole cluster, it handles worker node failures, replicating components, maintaining pods, etc. Controllers are responsible for overall status of entire cluster. It ensures that nodes are running all the time, correct number of pods are running as specified in config file.

Etcd is a data store that stores the cluster configuration in the format of key value format. It is developed by core OS. With the help of etcd file, it can restore entire cluster components from this stored cluster configuration. It is the

Fig 5: Kubernetes+Ceph+Rook integration

central database to store current cluster at any point of time. Etcd is a distributed key-value store where all configuration information is stored.

The **Worker node** are kind of virtual machine (VM's) running in cloud or on-prem, a physical server running inside data center. Physical machine capable of running container runtime can perform a worker node job. These nodes provide

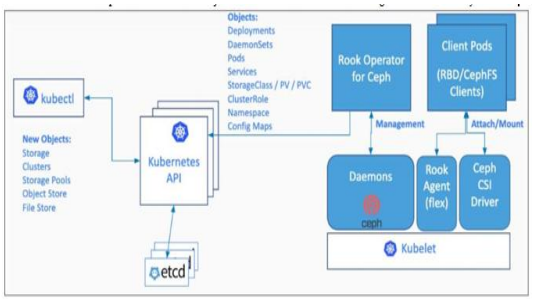

Figure 5: Kubernetes + Ceph with Rook architecture

underlying compute, storage and networking to the workloads. Together, these nodes form a cluster and run a workload assign to them by master node component as same as manager assign a task to individual team member. As master node using concept of controller can deploy pod in multiple pods to achieve fault-tolerance and replication.

Three main basic components of master node (control plane)

1. Kubelet
2. Kube-proxy
3. Container runtime

The **Kubelet** is the main service in worker which helps to communicate with API server and worker node.Once API server gets request from kubectl it will send request to schedule for workload on respective node using kubelet. NodeName with respective worker nodes will deploy pod on node and kubelet controller will get a notification from API server. It passes pod spec to container runtime to fetch image and run

The **Kube-proxy** load balances traffic between application workload. It is also called as service proxy which run on each node in the Kubernetes cluster& helps to establish communication between nodes & pod. Kube-proxy process

constantly check for services and perform rules on each node to forward traffic to back-end pods respectively.

Container runtime engine like Docker,rkt, containerdis installed on worker node which helps to pull and run images in pod. Once specification is defined in yaml file it looks for image section and fetch image from docker hub using container runtime engine

kubectl is a command line utility though which we can communicate or interact with master node to carry out specified job. Using kubectl utility we can control the Kubernetes cluster manager. There are two ways we can instruct the API server for CRUD operation in Kubernetes cluster using imperative way and a declarative way.

Rook is an open source cloud-native storage framework designed to manage storage solutions and is natively inte-grated with cloud-native environments like Openstack, Kubernetes. The main objective of Rook is to run File, Block and Object based storage systems into the Kubernetes cluster, where pods running inside node can consume the storage exposed using Rook. It runs as a Kubernetes operator, which help storage to run as software defined solution which can perform tasks like self-managing self-healing, and self-scaling service using Kubernetes objects. Applications run inside pod can mount block devices and file-system& use Amazon S3/Swift API for object storage managed by the Rook operator. The operators are deployed as pod, which automates and monitors the cluster to ensure the storage remains available and healthy. Inplace of deploying a new storage cluster, Rook helps to tune existing storage systems into cloud-native services on-top of Kubernetes. It uses Kubernetes object Custom Resource Definition (CRD) to manage Ceph storage. Rook is integrated with other database service like Cassandra DB, Cockroach DB for managing database services. Rook deployment supports several services for deploying the cluster using Prometheus and for dashboard using Grafana.

Ceph is distributed storage system that offers high performance, fault tolerance and redundancy. It manages distribution of copy across all nodes by copy-on write feature and no point of failure for storage system. The Ceph mainly support consistency, integrity and tolerance upon availability.

Ceph storage cluster is made up of several software daemons making sure all services associated with storage objects are running without any failure. A key element of Ceph is the implementation of a pseudo-

TEST Engineering & Management

May-June 2020
ISSN: 0193-4120 Page No. 5276-5281

random data distribution function (CRUSH) to determine where and how to store the date on the worker node. Ceph storage makes sure every data write operation an acknowledgment is sent to client only after all the replicas are written correctly

4. Kubernetes & Persistent Storage Test Infrastructure For Stateful Application

Dynamic provisioning of storage has been defined in storage class with the help of storage container drivers. Once storage classes are defined persistence volume of CephFS should be specified in yaml file. Persistence volume claim (PVC) should be notified so that other nodes workload does not use volume mapped to a pod. We have used Custom Resource Definition(CRD) for defining storage classes for Ceph.

Persistence Volume Claim(PVC) yaml file

```
apiVersion: ceph.rook.io/v1
kind: Persistent Volume Claim
metadata:
  name: sql-pvc
spec:
cephVersion: apps/v1
  image: ceph/ceph:v14.2
data Dir Host Path: /var/lib/rook
storage Class Name: rook-ceph-block
access Modes:
  - Read Write Once
  resources:
    requests:
      storage: 5Gi
nodename: node01
```

Above yaml file is run on master node which helps to integrate storage volume to nodes.

Test Setup

Hardware: Intel i7 8 Hyper-thread cores & 16 GB RAM
Operating System: - Ubuntu 18.04 LTS
VM WorkStation: - VirtualBox
No of VMs: - 4(1 Master & 3 Worker Node)
Hard Disk: - 100GB mounted to Nodes
Kubernetes v1.14, Docker v18.09.2

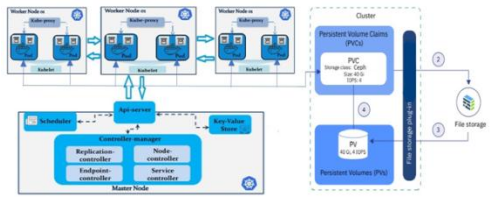

Figure 6: Kubernetes+Persistence Storage Test Implementation

RMS6: As we have used HDD disk for persistence storage on storing data of stateful application. We have measured disk performance for random read/write & sequential read/write Below are the test outcomes:

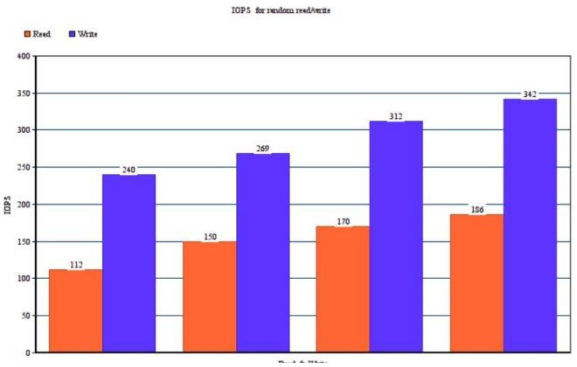

Figure 7: Persistent Drives IOPS Throughput Report

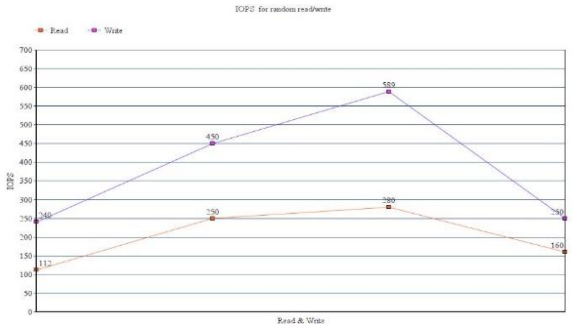

Figure 8:Persistent Drives IOPS Latency Report

5. Conclusion

This research work performed a preliminary investigation on running stateful application over containerized environment with persistent storage. We have successful run database service workload on Kubernetes with Ceph as storage and performed IOPS test to ensure workload are getting run in containerized environment without any issues. As a part of this research we have accomplished of integrating Kubernetes with persistent storage for running stateful workload.

References

[1] G. Tlili, M. F. Zhani and H. Elbiaze, ”On providing deadline-aware cloud storage services,” 21st Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN), Paris, France, 2018.

[2] J. Gantz, and D. Reinsel, The digital universe in 2020: Big data, bigger digital shadows, and biggest growth in the far east. IDC iView: IDC Analyze the future, 1-16, 2012.

[3] R. H□ppli, T. M. Bohnert, and L. Militano ”Hera Object Storage: A seamless, Automated Multi-Tiering Solution on Top of Openstack Swift”, 8th IEEE International Symposium on Cloud and Services Computing (SC2), 2018.

[4] Z., Ou, M., Song, Z. H., Hwang, A., Yl¨a-J¨a¨aski, R., Wang, Y., Cui, and P. Hui, ”Is cloud storage ready? Performance

May-June 2020
ISSN: 0193-4120 Page No. 5276-5281

comparison of representative IP-based storage systems", Journal of Systems and Software.

[5] C. Cerin, C. Coti, P. Delort, and F. Diaz, Downtime statistics of current cloud solutions, IWGCR: The International Working Group on Cloud Computing Resiliency, EU/USA, Tech. Rep. 001-en1-2013, Jun. 2013.

[6] [6] E. Brewer, "CAP twelve years later: How the" rules" have changed." Computer 45.2, pp. 23-29, 2012.

[7] C. Colman-Meixner, C. Develder, M. Tornatore and B. Mukherjee, "A Survey on Resiliency Techniques in Cloud Computing Infrastructures and Applications," in IEEE Communications Surveys & Tutorials, 2016.

[8] G. Toffetti, S. Brunner, M. Bl¨ochlinger, J. Spillner, and T. M. Bohnert, "Self-managing cloud-native applications: Design, implementation, and experience", in Future Generation Computer Systems, Vol. 72, 2017, pp. 165-179,

[9] A. Srbu, and O. Babaoglu, "Towards Data-Driven Autonomics in Data Centers", in International Conference on Cloud and Autonomic Computing, 2015.

[10] A. M. Kermarrec, E. Le Merrer, G. Straub, and A. Van Kempen, "Availability-based methods for distributed storage systems", in IEEE 31st Symposium on Reliable Distributed Systems (pp. 151-160), 2012.

[11] Chun, Byung-Gon and Dabek, Frank and Haeberlen, Andreas and Sit, Emil and Weatherspoon, Hakim and Kaashoek, M Frans and Kubiatowicz,John and Morris, Robert Tappan, "Efficient Replica Maintenance for Distributed Storage Systems", in NSDI, vol. 6, 2006.

[12] Zhu, Bingpeng and Wang, Gang and Liu, Xiaoguang and Hu, Dianming and Lin, Sheng and Ma, Jingwei, "Proactive drive failure prediction for large scale storage systems", in IEEE 29th Symposium on Mass Storage Systems and Technologies (MSST), 2013.

[13] I. a. c. Red Hat, "Ceph Documentation," 2018. [Online]. Available: http://docs.ceph.com/docs/master/. [Accessed 26 07 2018].

[14] The Kubernetes Authors, "What is Kubernetes?," The Linux Foundation, 2018. [Online]. Available: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/.