Running a CEPH-Cluster from a containerized infrastructure

Use case: mySQL-database

Julius Neudecker
Bachelor of Science
julius.neudecker@haw-hamburg.de

January 2020

Contents

Setting up and operating a storage cluster with high availability is a complex task. By using containerization, it is possible to abstract away and encapsulate some repetitive tasks. Therefore this study aims to analyse the possibility, implications and findings of a multi host CEPH-Cluster, where the individual daemons are entirely running within a containerized environment. To put the findings into a frame of reference, this study utilizes a mySQL-database which has special requirements on data storage. The key points in terms of advantages and disadvantages, data integrity, performance and administion are scrutinized. Major findings are that ... [insert findings here later] ... . Therefore running a distributed CEPH-storage in a containerization environment is ... [draw conclusion] ... .

1 Introduction

In times where information is a valuable asset, it is of paramount importance to have a scalable and reliable way of storing data and information. Considerations about data throughput and IOPS[1] are also a major design parameter on modern storage solutions. These different storage solutions provide different approaches on these considerations. For any given use case, there exist several options to adress these. Depending on the architecture and scope of the problem some are better suited than others. A few major considerations are apart from scalability, reliability and speed also cost effectiveness, vendor lock-in, complexity and granular customizability.

In general hardware based solutions have advantages in terms of raw performance but they often have significant disadvantages when it comes to vendor lock-in, easy scalability or restoration of corrupted disks. Software and network based solutions are *in principle* less performant. However this can be mitigated for the most part by scaling up.

Apart from propretary cloud storage providers i.e. AWS, Azure or Google, the **free market** [**correct term?**] is heavily dominated by CEPH by more than twice as much as the next competitor:

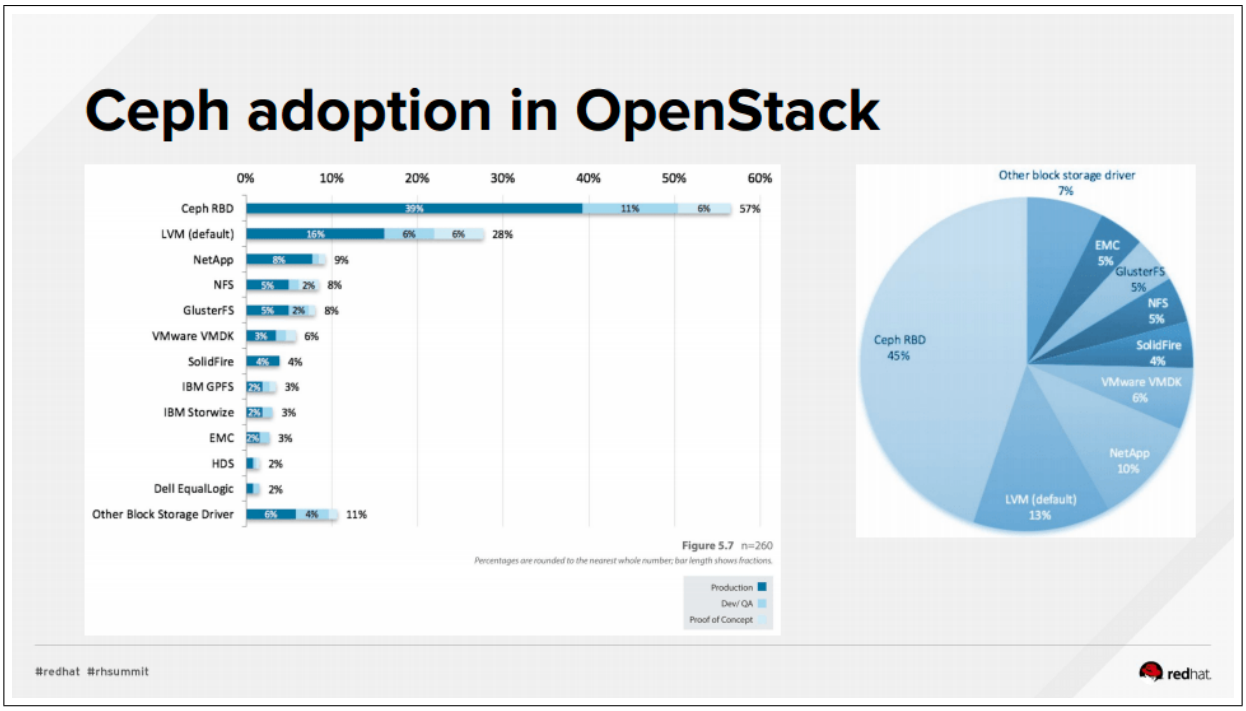

Figure 1: Adoption of CEPH in OpenStack in 2016, [**?**]

Being conceived by Sage Weil for his doctoral thesis [**?**], CEPH became part of the Linux Kernel in 2010 and was acquired by RedHat in 2014, CEPH is gaining popularity steadily since its introduction. **Source?**

Another modern important concept which is increasingly shaping modern technology stacks is *OS Level Virtualization* or *containerization* as its colloquially called. This way of deploying applications decreased the complexity, which is inherent to deploying several different applications to one single host machine. [**/Refactor this section /Since when?**]

Lastly, no storage solution exists without its use case. One major application, which requires flexible scaling are databases e.g. relational databases. To guarantee deterministic behaviour, database servers need to have specific properties to be suitable as database providers. The underlying storage solution is one of these.

[1] Input/Output Operations per Second

1.1 Problem domains

Managing a highly available storage cluster is not a trivial task. Apart from provisioning and monitoring the hardware, setting up multiple systems concurrently is a daunting task. Nowadays with infrastructure automation tools like Salt, Chef or Puppet, this is easier than ever. However, it can still be a tedious task to tweak the configuration of these tools in order to make it work on a complex or diverse infrastructure. Especially in times when upates and EOL[2] events create incompatibilites between working application stacks on any given host machine.

As for CEPH, this is especially true, since there are three major versions in production [**?**], as of early 2021. Deprecated features and bugfixes create functional inconsistencies between major versions, hence it is a necessity to keep a cluster in production up to date. This necessitates constant modification and testing of the previously mentioned automation tools.

One way to isolate these problems is *OS Level Virtualization*. By doing so, every application or application stack exists within a so called *Container* and is therefore isolated from the host to a certain degree. One inherent issue in this context is that they are stateless and ephemeral. This means that they can *by principle* be created and deleted according to momentary requirements. This process can be fully automated by means of using an *orchestration software*. Therefore the virtualized production environment has to be configured in a way that allows is to provide the stability which is required for a storage engine.

Trying to overcome the difficulties in setting up and manage a CEPH cluster with containerization might seem contradicting at first sight. The following sections focus on the major problem and adresses these. There are many related topics such as further optimizations for one or another particular use case. To adress these would go beyond the scope of this paper. Nevertheless a brief outlook on further considerations will be provided at the end of chapter 4.5.

1.2 Definition of research goal

The goal of this research is to evaluate if setting up a CEPH-storage cluster with containers is possible and if so, if it is a feasable option for a production environment. In order to make a conclusive assessment, three main points have to be examined. In order to draw a meaningful conclusion, these three main topics must be evaluated in contrast to a *non*-containerized cluster.

Data Integrity This means running a service on the cluster, which is very sensitive to data inconsistencies. In this particular example a mySQL database is chosen. As for reasons which will be discussed in chapter 3.4, Databases have some unique properties, which makes them more sensitive to issues with data replication and keeping clustered storages in sync. Therefore this use case is chosen as a suitable real world application.

Performance Since performance is a main consideration in production environments, this is next to data integrity the second most important concern. Depending on the overall cost structure of the environment, a performance penalty might outweigh the benefits. In this case a containerized cluster would be *technically* possible but not economically feasable. Depending on the use case the important metric also differs. Fileserving services like storage clouds or streaming services rely more on raw throughput. The data traffic with databases is generally rather small, therefore IOPS[3] are more important.

Administration One important reason to do reasearch in this topic is to evaluate if the time and effort to set up a cluster brings benefits in terms of administrative expenditure. At first sight a viable metric could be the spent time from starting to have a cluster up and running. However, depending on the production environment the results may vary to a wide degree. Therefore chapter 4.4 will try to generate more abstract metrics for an objective evaluation.

[2] End Of Life
[3] Input/Output Operations per Seconds