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:

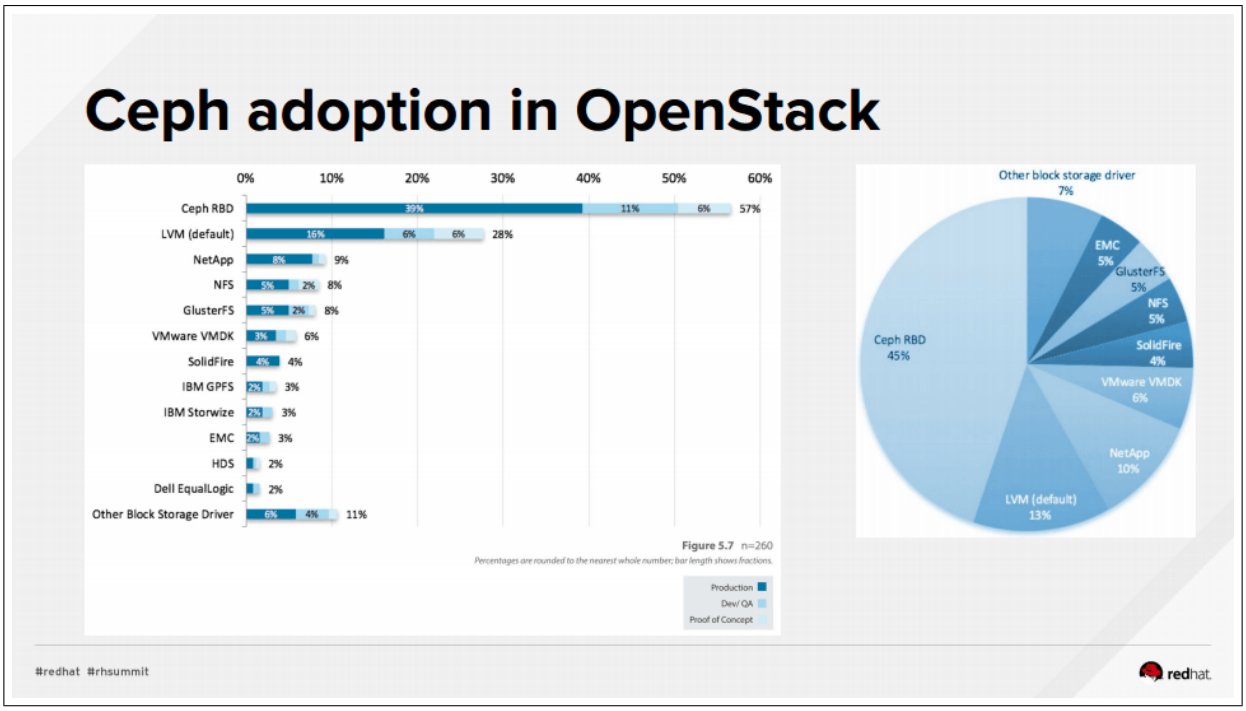

Figure 1: Adoption of CEPH in OpenStack in 2016, [1]

Being conceived by Sage Weil for his doctoral thesis [2], 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 [3], 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

1.3 Related Work

Since containerization and CEPH are already well established technologies, this section gives a brief overview of other research which was already conducted in this area. RedHat in cooperation with Percona and Supermicro already conducted extensive research wheter it is feasable to run a SQL-Database on a CEPH-Cluster [4]. They concluded that indeed running a database on an optimized CEPH-cluster exceeds industry standard database solutions. Furthermore scaling horizontally is easier with a CEPH Cluster. Nevertheless it should be mentioned, that in this study the Percona SQL distribution was utilized, which provides a native interface for kernel based RBD[4].

Hong et.al. used CEPH to create a framework which aims to mitigate handling issues with database containers. This includes the issues mentioned in chapter1.1 that containers aren't persistent ways to store data [5]. However this paper does not use a CEPH-cluster to host a database and has therefore only very little relevance to the topic of this paper.

Although there are numerous other papers which discuss the process of setting up CEPH itself or in the context of an OpenStack environment, none of them discussed a similar scope.

2 Setting up CEPH on Docker

This section is about setting up CEPH with containerization. Firstly, an extensive insight into CEPH and its underlying structure is provided. Secondly The system-architecture for the experiments in the scope of this paper is explained. This second section will make a small detour into containerization, what it means, what the benefits and the drawbacks are.

2.1 CEPH Architecture

In short CEPH is a distributed solution for storage clusters. On one side it is specifically tailored to provide maximum reliability by distributing data over different disks, machines or even datacenters and therefore also improving overall scalability and performance. On the other side it is also a cost effective because it is open source[**Github Link**] and runs on commodity hardware.

The general structure includes several services which manage different functions within the cluster. In the context of CEPH these are called deamons, which are discussed in detail in the following sections.

Because it is a distributed system and may span over several physical machines, racks or even datacenters, inter-daemon communication plays a key role. For the purpose of autonomous data replication the cluster relies on a dedicated network. This network is separated from the public network, which is for client data access. Figure 2 provides a general overview.

How the data is distributed internally is determined by two main factors: *Placement Groups* and the *CRUSH* algorithm. Placement Groups (abbreviated *PGs*) are a way to keep track of the physical location of objects. Files are split up into different objects, where each object is stored in a different PG. Which one is determined by the CRUSH[5] algorithm [2]. Figure 3 shows a schema of how this works. The reasoning behind PGs is that keeping track of millions of objects via metadata is computationally expensive [7]. Therefore Objects are hashed and assigned to PGs. A so called CRUSH-Map keeps track of the structure of the whole cluster. This will become important in section 2.2.4.

2.1.1 Cluster Access

To access the cluster ceph provides different interfaces depending on the use-case:

- *CephFS* - A POSIX[6] conform filesystem
- *LIBRBD/KRBD* - A block device

[4]Replicated Block Devices
[5]Controlled, Scalable, Decentralized Placement of Replicated Data
[6]Portable Operating System Interface - Part of the Unix specification

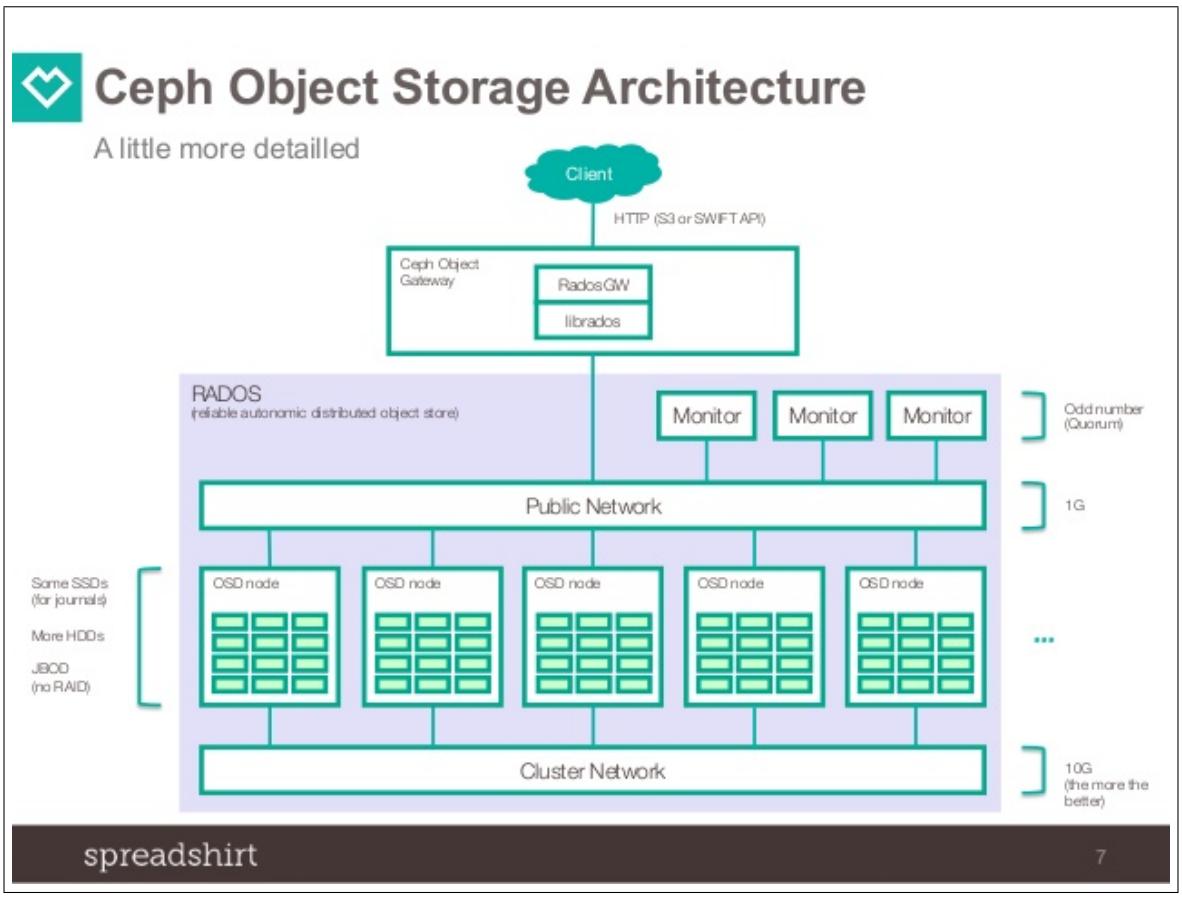

Figure 2: CEPH Architecture, [6]

- *RADOSGW* - An REST gateway for storage buckets
- *LIBRADOS* - The API to access the cluster directly from applications

All these interfaces are based on LIBRADOS, provides the interface to access RADOS. This is the underlying object store responsible for distributing the data over several phsyical disks. This is done by the *OSD*-Daemon.

CephFS is the filesystem, which allows for CEPH to behave to the user or the operating system like any other filesystem would. However is has to be distinguished from from block device. The key is that it is not a filesystem like EXT4 or NTFS. It is more a translation layer to the RADOS[7] interface, and provides features like snapshots. A required metadata service called *MDS* provides journaling functionality and handle multi user data access. Because the throughput of the whole cluster scales linearly with the size of the cluster, often several different MDSs are required to handle the load and distribute the file metadata mitigate single points of failure. Is is best practice store and server the MDS from fast solid state drives to migigate bottlenecks.

LIBRBD/KRBD is a virtual block device. The difference to CephFS is that is behaves like a physical disk. Therefore it can be formatted with any filesystem like EXT4 or NTFS. The distinction between *LIB*RBD and *K*RBD means userspace for LIBRBD and kernelspace for KRBD. To explain these differences between these two would go beyond the scope of this paper and requires knowledge about kernels, operating systems and memory handling.

RADOSGW provides as REST gateway access storage buckets. Unlike filesystems or block devices, these storage buckets don't have a hierarchical structure with directories. All objects are stored on

[7]Reliable Autonomous Distributed Object Storage

the same level, therefore the analogy to a bucket. This bucket is addressed by means of an API which provides also a metadata filter to select specific files directly. This way of storing data is also compatible to Amazons S3[8] product. A widely adopted usecase is serving static files on the internet for media content, javascript or css files by content delivery networks. They can also be used to store virtual disk images to be used in conjunction with RBDs.

LIBRADOS provides a native API to access RADOS directly without any file system or block device translation layer in between. The higher development costs for a more complex implementation might be worth the gains in throughput and IOPS for critical applications.

2.1.2 Object Storage Devices - OSD

Within the CEPH domain, these are the disks, where the actual data objects are stored to. One or more OSDs can be handled from one CEPH OSD Daemon, which provides the connecting layer to the whole cluster. Also one OSD can contain several Placement Groups. This depends on the configuration of the cluster. On which disk and in which PG and object is stored is determined by the CRUSH algorithm as shown in figure 3. Since data is replicated within the cluster and constantly updated, one OSD could fail and the data and PGs of this failed OSD will be written to another OSD. This constant data shuffling takes place transparent to the user and is the reason why CEPH needs a dedicated network as mentioned in section 2.1. How the data is physically stored on the disk differs between the two different backends used in older or newer releases.

2.1.3 Monitor Nodes - MON

Monitor Nodes (abbreviated as *MON*) have two main purposes. They maintain a copy of the cluster map to hand this out to a client, which connects to the cluster. Because of a fault this map might not reflect the state of the cluster accurately. Therefore an odd number of MON nodes have to agree on the current state of the cluster and distribute the correct cluster map. According to the CEPH documentation this is negotiated via the *PAXOS* algorithm [8]. Although a cluster could technically work with just one MON node, it is highly recommended to have at least three MON nodes, preventing a single point of failure.

2.1.4 Metadata Server - MDS

The metadata server is not neccessary for the cluster to operate as RBD, storage bucket or via Librados API. However to provide a journaled filesystem via the CephFS, this service is needed to provide the filesystem metadata such as permissions and timestamps. As stated from RedHat, it is best practice to deploy a MDS with big and fast nVME drives [4]. These can be also used in conjunction with memcached, where the recent metadata are stored into the RAM of the MDS. Overprovisioning, automatic failover and scalability are the main considerations for MDS to have high availability and high performance.

2.1.5 Manager - MGR

Since the 12.x release, CEPH needs a manager node to operate [10]. This manager does not provide any functionality to the cluster itself but acts as an interface to external monitoring tools and systems. Without it the state of the cluster won't appear to be healthy nor be externally visibly updated [10].

[8]Simple Storage Service

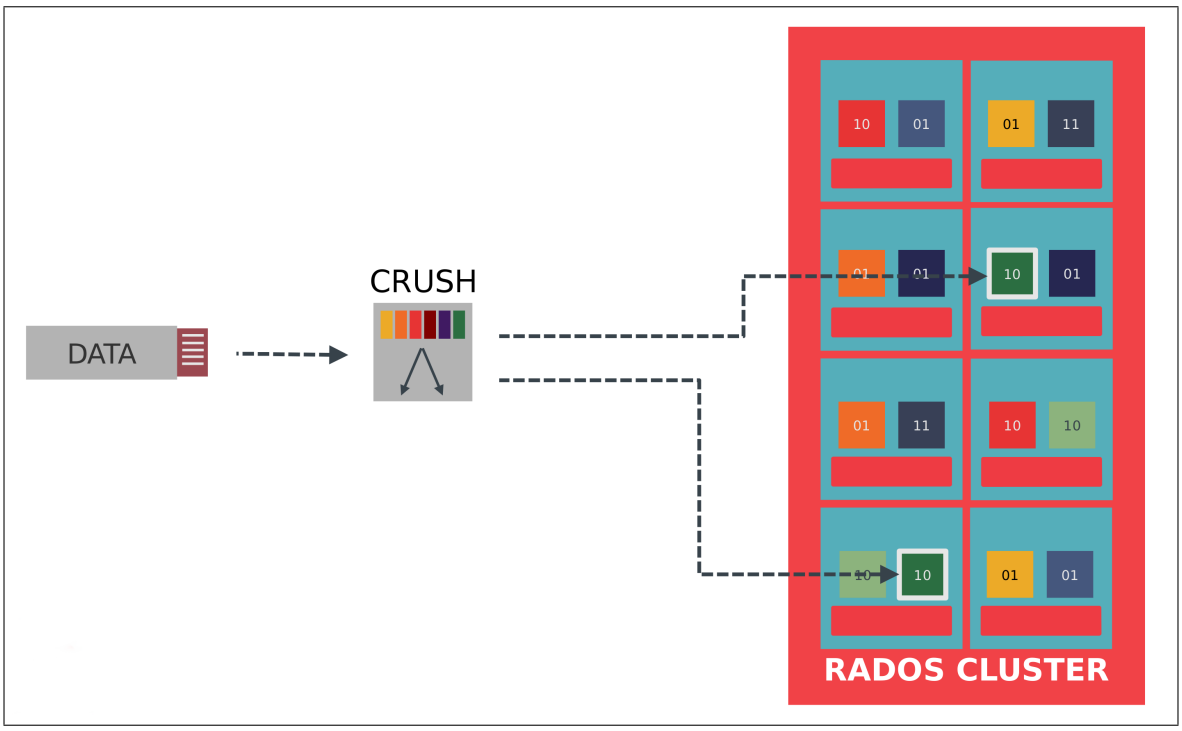

Figure 3: Placement Groups and CRUSH Algorithm, [9]

2.2 System Architecture

2.2.1 Containerization

2.2.2 Docker Config for CEPH Image

2.2.3 Orchestration with Kubernetes (or Docker Swarm maybe...)

2.2.4 CRUSH Fail mode

2.2.5 Issue with Docker Image

3 Database considerations

3.1 Databases

3.2 Architecture of mySQL

3.3 ACID

3.4 Problems with clusters

3.5 Considerations for this research

4 System Analysis

4.1 Disclaimer

[Bc of Corona I can't use lab therefore only my setup. Discuss some shortcomings and implications.]

4.2 Data Integrity

4.3 Performance Penalty

4.4 Administration

4.5 Tuning

[very briefly]

5 Conclusion

5.1 Advantages

5.2 Disadvantages

5.3 Performance

5.4 In Summary

References

[1] [Online]. Available: https://i2.wp.com/svbtleusercontent.com/mqwrzstn0uje0q.png?ssl=1

[2] S. A. Weil, "Ceph: reliable, scalable, and high-performance distributed storage," Ph.D. dissertation, University of California, Santa Cruz, 2007.

[3] "Ceph releases (index)." [Online]. Available: https://docs.ceph.com/en/latest/releases/

[4] "Deploying mysql databases on red hat ceph storage." [Online]. Available: https://www.redhat.com/de/resources/mysql-databases-ceph-storage-reference-architecture

[5] S. Hong, D. Li, and X. Huang, "Database docker persistence framework based on swarm and ceph," in *Proceedings of the 2019 3rd High Performance Computing and Cluster Technologies Conference*, 2019, pp. 249–253.

[6] J. Hadlich, "Ceph object storage at spreadshirt (july 2015, ceph berlin meetup)," Jul 2015. [Online]. Available: https://www.slideshare.net/jenshadlich/ceph-object-storage-at-spreadshirt-july-2015-ceph-berlin-meetup

[7] "Placement groups¶." [Online]. Available: https://docs.ceph.com/en/latest/rados/operations/placement-groups/#:~:text=Aplacementgroup(PG)aggregates,onaper-objectbasis.

[8] "Architecture¶." [Online]. Available: https://docs.ceph.com/en/latest/architecture/#high-availability-monitors

[9] "Ceph." [Online]. Available: https://www.thomas-krenn.com/de/wiki/Ceph

[10] "Ceph manager daemon¶." [Online]. Available: https://docs.ceph.com/en/latest/mgr/