https://doi.org/10.1051/epjconf/202429501021

Enabling Storage Business Continuity and Disaster Recovery with Ceph distributed storage

Enrico Bocchi[1]*, *Abhishek* Lekshmanan[1], *Roberto* Valverde[1], and *Zachary* Goggin[1]

[1]CERN, Esplanade des Particules 1, 1211 Geneva 23, Switzerland

Abstract. The Storage Group in the CERN IT Department operates several Ceph storage clusters with an overall capacity exceeding 100 PB. Ceph is a crucial component of the infrastructure delivering IT services to all the users of the Organization as it provides: i) Block storage for OpenStack, ii) CephFS, used as persistent storage by containers (OpenShift and Kubernetes) and as shared filesystems by HPC clusters and iii) S3 object storage for cloud-native applications, monitoring and software distribution across the WLCG.
The Ceph infrastructure at CERN is being rationalized and restructured to allow for the implementation of a Business Continuity / Disaster Recovery plan. In this paper, we give an overview of how we transitioned from a single cluster providing block storage to multiple ones, enabling Storage Availability zones, and how block storage backups can be achieved. We also illustrate future plans for file systems backups through `cback`, a `restic`-based scalable orchestrator, and how S3 implements data immutability and provides a highly available, Multi-Data Centre object storage service.

1 Introduction

Ceph is a cornerstone of CERN's IT infrastructure. It provides storage to services covering the needs of the whole Laboratory: Code repositories, monitoring and analytics services, documents and web hosting, and groupware. Over the years [1], it has demonstrated to be reliable, flexible, and capable of growing with capacity demand and evolving use cases.

Ceph is tightly integrated with our OpenStack-based cloud: CERN users submit a request for computing and storage quota and then administer their resources in a self-service fashion. This model of provisioning computing resources has proven to be very successful and greatly simplifies operation of storage for core IT and LHC services.

1.1 Introduction to Ceph

Maintaining access to and ensuring consistency of data is critical to what is expected of a storage system. Ceph [2] provides excellent reliability and performance, coupled with scalability. Reliability and scalability are achieved with the CRUSH algorithm [3], which ensures the distribution of data to underlying storage across failure domains, while not depending on a central namespace lookup. Based upon Reliable Autonomic Distributed Object Store (RADOS) [4], which for applications appears as a single logical object store, a Ceph cluster

*e-mail: enrico.bocchi@cern.ch

© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

ensures strong consistency of its stored objects. Internally RADOS stores everything as objects in a flat namespace, and both servers and clients can compute the location of an object without a single point of contention for lookup using the CRUSH algorithm.

Clients can access a Ceph cluster using 3 commonly used protocols: i) as virtual blocks via RADOS Block Device (RBD), ii) via HTTP using the Amazon S3 and OpenStack SWIFT API via RADOS Gateway (RGW), as well as iii) through a file system interface with Ceph File System (CephFS). Data is always stored as RADOS objects, however, different access protocols operate in different data pools and namespaces, e.g., one cluster will have a dedicated RADOS pool to store S3 objects and a second independent pool to store RBD images. Moreover, protocols are built with different transaction semantics, hence a file stored, e.g., via CephFS, cannot nominally be accessed over HTTP via RGW.

1.2 Business Continuity and Disaster Recovery Concepts

Business Continuity (BC) is largely an organization's ability to continue essential operations during a fault. According to the criticality of affected functions, one may want to put in place different strategies for the underlying IT services: i) Active-active setups are most suitable for mission-critical applications where higher cost and implementation complexity are warranted by the requirement of minimal downtime; ii) Warm standby and Pilot light are adequate for services that may run at limited capacity upon disruption and subsequently be scaled out to full capacity; iii) Cold restores from backups should be foreseen where higher downtime is acceptable or the nature of the adverse condition does not allow using other strategies.

In addition to BC, having a Disaster Recovery (DR) plan is essential in preparing for and recovering from unexpected disasters, e.g., human error operating an IT system, the deployment of buggy application, or ransomware attack [5]. It would be expected that DR defines cold restore as the minimal requirement: When facing scenarios leading to the loss of production data (e.g., a catastrophic software bug), all live replicas are likely to be affected and recovery from offline backups remains the only viable solution. Other strategies (i.e., active-active) would yield a faster recovery, but may not be executable under these circumstances.

2 Ceph as Infrastructure Component for Storage BC/DR

2.1 Problem Statement

The multiplicity of access protocols provided by Ceph (Sec. 1.1) has implications on the design of solutions for BC/DR: Each overlaying service, i.e., RBD, RGW, CephFS, has its own way of achieving active-active / active-passive setups or implementing backups. This is due to the fact that the overlaying service has better knowledge of what/when to replicate. For example, uploading an S3 object involves splitting it into chunks, writing the tail chunk, updating the head chunk, and finally creating an index entry. Replicating the object implies similar actions have to be replayed on the target cluster; This is easier to achieve through RGWs, rather than tracking and coordinating transactions at the lower RADOS layer.

Also, different BC/DR requirements imply different technical implementations. For example, providing a multi Data Centre (DC) active-active storage service implies distributing storage in two geographically separated DCs, applying every write transaction to all existing replicas, and ensuring that enough replicas would be available in the event of a DC loss. On the other hand, performing backups prioritizes the need for snapshot-like functionalities (i.e., an immutable view of a storage resource at a given point in time) and efficient ways to copy data to the target cluster without noticeable performance penalties on the live storage system.

As such, the complexity of achieving Ceph BC/DR derives from the large number of combinations of BC/DR strategies and Ceph access methods: No one catch-all solution exists, but rather separate ones have to be implemented for RBD, S3 objects, and CephFS. The desirable integration of BC/DR functionalities with OpenStack brings further constraints and requires additional implementation efforts.

2.2 Ceph RBDs for OpenStack

RBD allows storing block-based application data by chunking it and storing the resulting objects on RADOS. An RBD image is the virtualized equivalent of a physical disk, which can be consumed by clients via, e.g., kernel clients, `qemu/kvm`, and `iSCSI` Gateway. At CERN, RBDs back the whole OpenStack infrastructure as persistent storage attached to VMs.

2.2.1 Introducing Storage Availability Zones

To reduce the business impact of one RBD cluster being unavailable, new RBD clusters have been recently instantiated and grouped according to their quality of service (QoS). At the time of writing, RBD images are hosted on 5 clusters: 3 are interchangeable for general-purpose storage, 1 features full-flash devices for high-IO, low-latency workloads, and the last is hosted on a diesel-backed power supply, meant for critical applications. The clusters do not share any component of the Ceph control plane, are physically installed in different DC rooms, and are connected to a distinct network branch and UPS power feed.

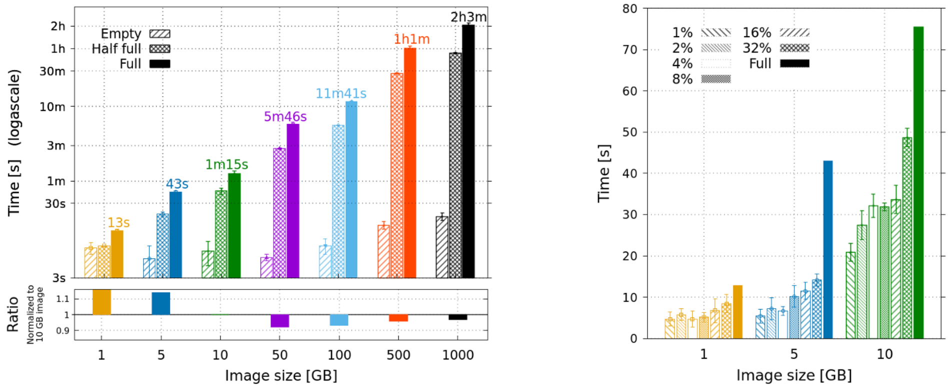
While the flash-based and the diesel-backed clusters are identifiable with clear volume type labels on OpenStack Cinder (i.e., `io` and `critical`, respectively), the 3 remaining general-purpose clusters should not be exposed with different types as they provide identical QoS. For this reason, we have introduced the concept of *Storage Availability Zones*: The 3 clusters are grouped under the Cinder type `standard`, and the choice of the underlying cluster to use can be made by the user explicitly, specifying the desired availability zone, or delegated to Cinder, which picks the best cluster according to an internal weighting function.

2.2.2 RBD Backups with OpenStack Cinder

Cinder implements a backup functionality [6] for block storage, whose backups can be stored on various storage backends. When used in combination with Ceph RBD, Cinder automates taking a snapshot of the source image, allowing for point-in-time consistent backups.

S3 would be our preferred target to store RBD backups as it would enable storing backups also on external clouds. Unfortunately, its implementation is quite inefficient: The backup driver sequentially reads chunks of the source image, applies compression, and finally uploads the resulting object to S3. This leads to poor performance, taking 80 seconds to backup a 1 GB image (7 minutes for 5 GB). Support for incremental backups is also poor, as the driver reads the entirety of the source image and of the previous backup to identify deltas. Several parameters to tune object sizes and switch among compression algorithms exist, but they all come with a performance or space consumption tradeoff: One has to choose between fast (i.e., no compression and large object size to profit from multipart concurrent uploads to S3) and space-efficient backups (i.e., compressed incremental with smaller S3 object size).

A very effective alternative implementation of backups exists using Ceph RBD both as source and as target storage. In this scenario, the OpenStack Cinder driver leverages `librbd`, which implements native functions to identify changes and copy them over to the target. Fig. 1a shows the time taken to backup RBD images being empty (dashed bars, left-hand side), 50% full (checkered bars, middle), or full (solid bars, right-hand side), with various



(a) Full backup with images ranging from 1 GB to 1 TB. The lower part shows the time ratio w.r.t. a 10 GB image.

(b) Incremental backup changing a given percentage of the source image.

Figure 1: Time to backup RBD images using OpenStack Cinder and Ceph RBD as target.

sizes between 1 GB and 1 TB. One can observe the time taken is proportional to the size of the image (the lower part of the plot highlights the ratio of time using the 10 GB image backup as reference) and to the volume of data stored in it. Also, the backup of empty images always finishes within seconds (≤ 20 s) independently of their size: Images are thinly provisioned and `librbd` is capable of detecting 0-extents through `object-map` [7]. Lastly, for incremental backups, a fresh snapshot of the source image is created and compared to the state of the image on the target. This allows for the precise detection of modified extents, which are then copied to the target cluster. Fig.1b shows the time for incremental backups when changing a controlled amount of data (1% – 32% of the original bytes): While the time is not perfectly proportional, the trend shows a clear correlation between changes and backup duration, which is also reflected in the amount of bytes transferred between source and target clusters.

2.3 S3 Object Storage

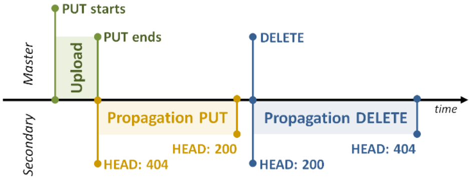
The RADOS Gateway (RGW) is a storage interface exposing S3 and SWIFT APIs on top of RADOS. RGWs expose users, buckets, and other S3 functionality and heavily utilize object metadata for implementing bucket indexes and ACLs. At CERN, S3 is used by many cloud-native applications and regular workloads requiring an object store.

2.3.1 Multisite Concepts

Since the Jewel release of Ceph (v10.2.0 - 2016), RGW introduced the concept of *multisite*, i.e., an active-active multi-zone object storage service. A *zone* refers to a logical group of one or more RGW instances; All RGWs in one zone serve objects from the same Ceph pool. A *zonegroup*, instead, is a collection of zones that usually span multiple clusters and/or geographies. Every zonegroup requires a master zone to be configured.

RGW primarily deals with 2 categories of data: The actual data, which is the objects themselves, and the associated metadata, which includes, e.g., the user accounts and bucket names. Metadata is shared across zonegroups, while data is shared within a zonegroup, across zones. The need for metadata sharing arises from the fact that bucket names need to be globally unique across all zonegroups, as the S3 API allows for buckets to be addressed with virtual-hosted–style URLs (i.e., `mybucket.s3.cern.ch`). Additionally, a system spanning across geographies will still want to use the same accounts, so user accounts need to be shared as well. The master zone of the master zonegroup will be the source of truth for all metadata: Any bucket/user/account creation or modification goes through this zone.

2.3.2 Traditional Multisite Tests



(a) Schematic of the Multisite testbed.

	PUT	Propagation PUT	Propagation DELETE
Mean	1.15	7.75	10.80
StDev	0.01	2.70	2.00

(b) Duration of various operations (in seconds).

Figure 2: Testbed and measurements of multisite propagation delays.

With traditional multisite configuration [8], data is mirrored across all zones. Synchronization across zones is performed by RGWs by maintaining indexed logs of operations that are shipped across and replayed. Writes are strongly consistent with the zone the client writes to and eventually consistent with the other zones. Due to the eventual nature of remote zone replication, we measured how long `PUT` and `DELETE` requests take to propagate to a remote zone using objects of 4 kB, 1 MB, 5 MB, and 10 MB. The testbed implemented to perform such measurements is described in Fig. 2a, while measured delays are reported in Tab. 2b.

Propagation delays are largely not dependent on the object sizes (which are small, and network bandwidth from the testbed to the S3 cluster is plentiful). More interestingly, `DELETEs` take longer to propagate than `PUTs`: In an eventual system, a create followed by an immediate delete will have various states of replication in the remote zones, i.e., a create may or may not have been applied yet and delete should only be applied in order. This problem can be solved via deletion tombstones such that, when the remote zone has not captured the `PUT`, both `PUT` and `DELETE` on the same object can be squashed to a `no-op`. However, in case the `DELETE` does not propagate in the same log segment, the `PUT` has to be replayed before. Only when all the zones have caught up with the logs, can the object actually be deleted.

2.3.3 Immutable Objects for Backups on S3

S3 objects are by nature immutable, so a modification to an object consists of a full rewrite. In compliance and data protection scenarios, it is desirable to prevent rewrites and be able to recover older versions when required. Similar functionalities are also valuable when recovering from a ransomware attack. RGW provides functionalities including:

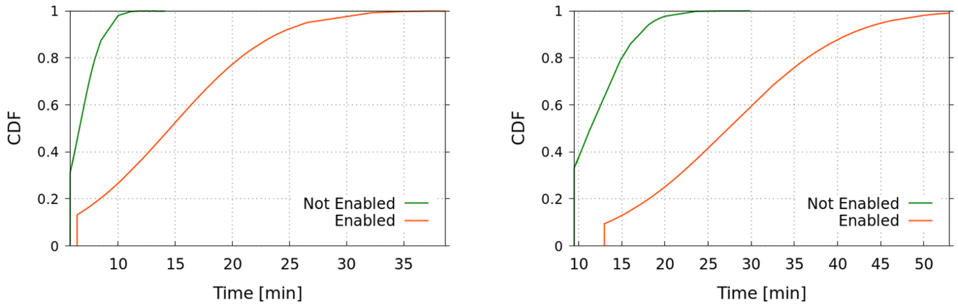
- **Object Versioning**: If enabled, object re-uploads will cause the existing object to be stored as a version, which can be downloaded or deleted by specifying the version identifier. To avoid bloating used capacity, non-current versions beyond a particular amount of days can be automatically deleted by the system with bucket lifecycle policies [9].
- **Object Lock**: The idea is to set a policy that would prevent overwrites/deletes of objects. Typically, a fixed retention period can be configured for the object lock, so to re-enable

all actions when the lock expires. Retention modes control the behavior of locked objects and include i) Governance mode, allowing the modification of lock settings only by special users, ii) Compliance mode, which applies a stricter policy preventing all users (including the bucket owner) from modifying any lock settings, and iii) Legal hold, which enforces an indefinite retention period on an object until the Legal hold has been removed.

2.4 Ceph File System

The Ceph File System (CephFS) is a modern POSIX-compliant file system that can act as a file store for home directories or networked, shared storage for distributed workflows and applications. CephFS at CERN is largely used by High Performance Computing (HPC) workloads as scratch space and container-based applications running on Kubernetes and OpenShift clusters. It is generally provisioned via OpenStack Manila [10].

2.4.1 CephFS Snapshots and Mirroring



(a) CDFs comparing the time to `untar` the Linux Kernel archive with and without snapshots.

(b) CDFs comparing the time to delete files previously extracted from the Linux Kernel archive.

Figure 3: CDFs comparing the distribution (200 repetitions) of the time taken to inflate the Linux Kernel archive and to remove the resulting files. The skew between green and red lines indicates the performance penalty introduced when using snapshots.

CephFS has implemented snapshots and mirroring feature sets: Snapshots are an immutable view of the file system (or one of its subtrees) at a specific point in time, while mirroring allows for the asynchronous replication of snapshots to a target CephFS by means of the `cephfs-mirror` tool. Snapshots are implemented as an asynchronous copy-on-write operation, while their creation (or deletion) can be achieved simply by creating (or deleting) a folder contained within a special purpose, reserved name directory called `.snap`.

Unfortunately, the use of snapshots severely degrades the performance of CephFS. Fig. 3 shows the Cumulative Distribution Functions (CDFs) resulting from 200 repetitions of inflating the Linux Kernel archive and deleting the produced files. In the case of inflating (Fig. 3a), 95% (50%) of runs complete in less than 9 minutes (6m 40s) with snapshots not enabled. Instead, the distribution of events is more dispersed with snapshots enabled, where 95% of runs take up to 26m 30s. Similar observations hold true also for the removal of files (Fig. 3b).

When working with large (i.e., PB-scale) and busy (i.e., thousands of metadata operations per second) clusters, such as the ones in production at CERN, enabling snapshots may jeopardize performance and make the service inadequate for more demanding workloads. For this reason, the use of snapshots is not considered viable at the moment. As a consequence, the mirroring functionality (which depends on snapshots) has not been evaluated.

2.4.2 CephFS Backups with cback

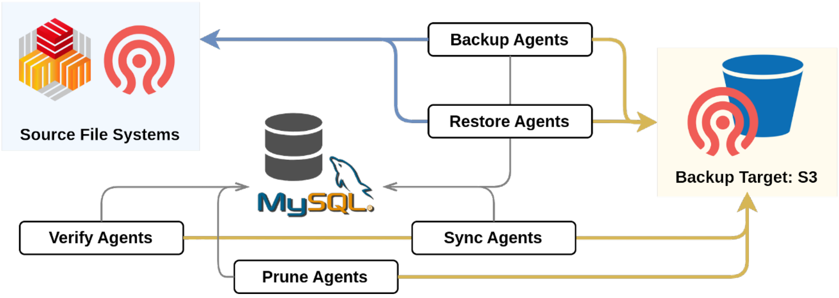


Figure 4: `cback` architecture. A central database stores the state of jobs (e.g., backup, restore, prune), which are then executed asynchronously by specialized Restic agents.

CERN creates backups for other file system by means of `cback` [11], an in-house orchestrator that builds on `restic` by adding horizontally-scalable agents and a centralized queue to track the job status (Fig. 4). `cback` can back up virtually any mounted file system to S3. During the backup process, space-saving methods (e.g., deduplication, compression) and encryption are applied prior to uploading to S3. Incremental backups are also supported.

Given that the performance overhead for CephFS snapshots has been deemed unworkable (Sec. 2.4.1), the existing `cback` infrastructure has been tested also for CephFS backups. The results have been very positive so far and there are plans to expand the utilization of `cback` by making it a self-service backup solution with OpenStack Manila.

The current implementation, however, suffers from the limitation of not guaranteeing point-in-time consistency: `cback` will back up a live file system whose files may change while the back up is in progress. Point-in-time consistency is only achievable if the source file system provides a snapshot-like functionality. While there are no immediate plans to use CephFS snapshots, `cback` would be able to use those as sources seamlessly. Having CephFS snapshots remains a requirement to achieve a complete backup solution.

3 Conclusions and Future Work

Ceph provides a remarkable set of capabilities allowing active-active, active-passive, and backup-restore for block, object, and file storage. While relevant for the implementation of a BC/DR strategy, each of these capabilities is tailored to a specific choice of BC/DR policy and Ceph storage type, implying the cost of operating one dedicated solution for each desired combination. In addition, some of these capabilities have not reached the expected level of maturity to enable them on large-scale production clusters, causing severe performance degradation or missing integration with overlaying OpenStack.

Block storage based on Ceph RBD takes advantage of being a simple though refined technology with a fully-fledged feature set: Snapshotting and replication represent the means with which RBD images can be efficiently replicated and used in active-passive or backup-restore scenarios. Also, the integration with OpenStack Cinder grants additional flexibility if such features should be exposed to end users.

Replication capabilities of S3 object storage primarily address the realization of high(er) availability, rather than active-passive or backup. Multisite consistency, however, has to account for a non-negligible replication delay, which may be undesirable (or unacceptable) for certain applications. S3 also provides useful features to achieve data immutability, making it a good candidate as the backup target for other storages.

CephFS is the least mature technology in BC/DR terms: While implementing valuable snapshotting functionality, this demonstrated a severe impact on performance, making it less attractive for large-scale clusters. In addition, replication capabilities provided by mirroring may not satisfy all use cases and are not integrated with OpenStack Manila, at the time of writing. Backing up CephFS with `cback` is an option that will be further explored, though it comes at the cost of increasing the load on central metadata servers, as the tool will have to iterate over the directory tree to identify changes and new files.

At the time of writing, the BC/DR policy is being defined and will then be reflected in technical solutions to implement. One could foresee having backups enabled for all types of storage as the next step to reach production: RBD snapshots coupled with Cinder backups is the most effective and straight-forward solution for block storage, S3 objects versioning and immutability can be of help for storing immutable backups, while `cback` can be used as a stopgap solution before the performance penalty introduced by CephFS snapshots is addressed. Additional solutions for high(er)-availability, e.g., S3 multisite or multi-DC clusters, will be deployed at a later stage if there is a need from higher-level applications.

References

[1] D. van der Ster, A. Wiebalck, *Building an organic block storage service at CERN with Ceph*, Journal of Physics: Conference Series **513**, 042047 (2014)

[2] S.A. Weil, S.A. Brandt, E.L. Miller, D.D.E. Long, C. Maltzahn, *Ceph: A Scalable, High-Performance Distributed File System*, in *Proceedings of the 7th Symposium on Operating Systems Design and Implementation* (USENIX Association, USA, 2006), OSDI ’06, p. 307–320, ISBN 1931971471

[3] S.A. Weil, S.A. Brandt, E.L. Miller, C. Maltzahn, *CRUSH: Controlled, Scalable, Decentralized Placement of Replicated Data*, in *SC ’06: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing* (2006), pp. 31–31

[4] S.A. Weil, A.W. Leung, S.A. Brandt, C. Maltzahn, *RADOS: A Scalable, Reliable Storage Service for Petabyte-Scale Storage Clusters*, in *Proceedings of the 2nd International Workshop on Petascale Data Storage: Held in Conjunction with Supercomputing ’07* (Association for Computing Machinery, New York, NY, USA, 2007), PDSW ’07, p. 35–44, ISBN 9781595938992, `https://doi.org/10.1145/1374596.1374606`

[5] K. Krishnan, *Weathering the unexpected: Failures happen, and resilience drills help organizations prepare for them.*, Queue **10**, 30–37 (2012)

[6] *OpenStack Cinder Backups*, Cinder Volume Backups, accessed 2023-12-18

[7] *Ceph enable the object map feature*, Ceph Blog (2015), accessed 2023-12-18

[8] *Ceph Rados Gateway Multi-Site*, Ceph Docs – Multi-Site, accessed 2023-12-18

[9] *Ceph S3 Bucket Policies*, Ceph Docs – Bucket Policies, accessed 2023-12-18

[10] *OpenStack Manila*, Manila Project Wiki, accessed 2023-12-18

[11] R. Valverde Cameselle, H. Gonzalez Labrador, *Addressing a billion-entries multi-petabyte distributed file system backup problem with cback: from files to objects*, EPJ Web Conf. **251**, 02071 (2021)