Addressing a billion-entries multi-petabyte distributed file system backup problem with *cback* : from files to objects

Roberto Valverde Cameselle[1,*] and *Hugo* Gonzalez Labrador[1,**]
[1]CERN - European Organization for Nuclear Research, 1211 Geneva, CH

Abstract. CERNBox is the cloud collaboration hub at CERN. The service has more than 37,000 user accounts. The backup of user and project spaces data is critical for the service. The underlying storage system hosts over a billion files which amount to 12PB of storage distributed over thousands of disks with a two-replica layout. Performing a backup operation over this vast amount of data and number of files is a non-trivial task. The original CERNBox backup system (an in-house event-driven file-level system) has been reconsidered and replaced by a new distributed and scalable backup infrastructure based on the open source tool RESTIC. The new system, codenamed *cback*, provides features needed in the HEP community to guarantee data safety and smooth operation from the system administrators. Daily snapshot-based backups of all our user and project areas along with automatic verification and restores are possible with this the new development. The backup data is also de-duplicated in blocks and stored as objects in a disk-based S3 cluster in another geographical location on the CERN campus, reducing storage costs and protecting critical data from major catastrophic events. We report on the design and operational experience of running the system and future improvement possibilities.

1 Introduction

At CERN, the Storage Group provides daily backups for all data stored in CERNBox[1–3]. The existing event-driven backup system has been running successfully since years but the lack of features (not snapshot based, not automatic verification, not resilient against disaster scenarios,...) and operational costs triggered the kick-start of a new project in 2019 to rethink our backup strategy towards a more feature-rich (snapshot based, scalable, backup verification, different geographical location,...) and cost effective solution.

CERNBox currently hosts over 1.6 billion files and 12PB of storage data distributed in 8 clusters, 93 servers and 4400 hard disks. Each file is stored in 2-replica layout across two different servers. When the data to be backed up is in the order of billions of files, at Petabyte volume scale, the process of having daily snapshot-based backups turns quickly into a challenge. The time factor is crucial and techniques as backup parallelization have to be explored to optimize the available time. Other important factor is minimising the footprint of the backup in terms of space occupancy to reduce the cost of the underlying infrastructure.

[*]e-mail: roberto.valverde.cameselle@cern.ch
[**]e-mail: hugo.gonzalez.labrador@cern.ch

Strategies like file de-duplication and efficient disk layouts have therefore to be considered. Other factors like reducing the impact of the backup process in the running service cannot be overseen. A clear and revised disaster recovery plan together with a reliable and verifiable backup system is an important part of the service. In this study we introduce an example of how to address this challenge using a backup orchestration system based on open source technology which can be integrated in dynamic and complex storage environments.

A backup system is more than merely copying data from one place to another and therefore proper analysis is needed as for the service itself. In the process of finding the most suitable backup solution, the following aspects should be taken into consideration.

2 Backup Analysis

The storage of the backup data requires resources and depending of its size, this could increase the cost exponentially. It is important to backup the essential data for the service and to filter out data that is not relevant or that can be recovered from other sources.

While deciding about the placement of the backup data, the main underlying question should be "Is the backup data safe in case of a disaster?". Storing the backup in the same physical place of the original data would not help in case of flooding or a fire. For critical backups, more than one backup can be considered, ideally stored in different geographical locations. Software disaster and bugs in storage systems could cause irreversible data corruption and need to be taken into consideration, specially if source data and backup data are stored using the same technology.

To reduce the storage footprint of the backup data a retention policy could be put in place in order to delete the old backups. This policy needs to be agreed within the organization taking into account the storage costs and the particular usage of the data. In the case of backups that involve personal data, a fine grained deletion may be required by data protection laws. One of the questions that could be raised to define the retention policy could be "is still useful a backup made 1 year ago?".

A good practise of managing a backup systems is to perform periodical backup verification. Scheduled verification procedures that involve actual file restoration give information about the correctness of the stored data and can be used to measure the time needed to restore. Sporadic human verification will help to validate the automatic verification.

If there are expectations that the restore of the backup will be exercised regularly, having automation tools in place could help to perform these tasks easily and, if required, in an unattended way. Exposing the restore functionality to the end-users in a self-service approach reduces the operational cost.

A backup system should cope with the growing trends of the service. If the service is ready to scale, the backup system should be able to scale as well.

Nowadays, a great effort is taken to protect the services and applications against security incidents (targeted attacks, data leaks, ...). The same principles can be applied to the security of the backups. Mechanisms such as encryption ensures that the data is not readable in case it was compromised.

Last but not least, it is important to evaluate the risks of executing the backup on running services and to try to identify and minimize potential impacts on the users experience.

2.1 CERNBox and EOS

CERNBox is a cloud collaboration hub for CERN community: it allows syncing and sharing files on all major mobile and desktop platforms (Linux, Windows, Mac OS, Android, iOS) aiming to provide offline availability for the data stored in the CERN EOS [4, 5] infrastructure. In addition, it satisfies the high demand in the community for an easily accessible cloud storage solution, an alternative to popular services like Dropbox or Google Drive. Integration of the CERNBox service with the EOS storage back-end was a step forward for providing synchronization and sharing capabilities for scientific and engineering use cases. CERNBox provides the integration and sharing capabilities with the LHC data analysis tools and transfer services; promoting a new integrated way of accessing data for scientific research. The core of the service is a sharing system based on components provided by the ownCloud[1] open software stack. CERNBox provides a service layer on top of storage systems already provided by the CERN Storage group and offers various modern ways to access data on cloud storage (see Figure 1):

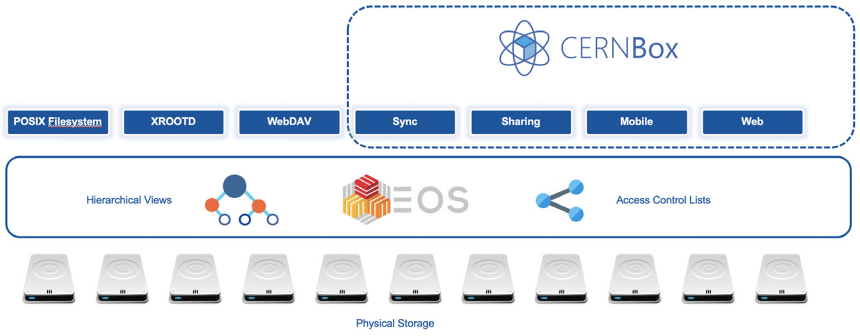


Figure 1. CERNBox Architecture

As depicted in figure 1, CERNBox allows accessing files via a Sync client software running on PCs and Macs, mobile application for portable devices (Android, iOS) as well as through the web interface. Files can be also accessed via the protocols already provided by EOS (WebDAV, XRootD). Permissions are propagated from the ownCloud system to the EOS access control list (ACL) stored as metadata attributes.

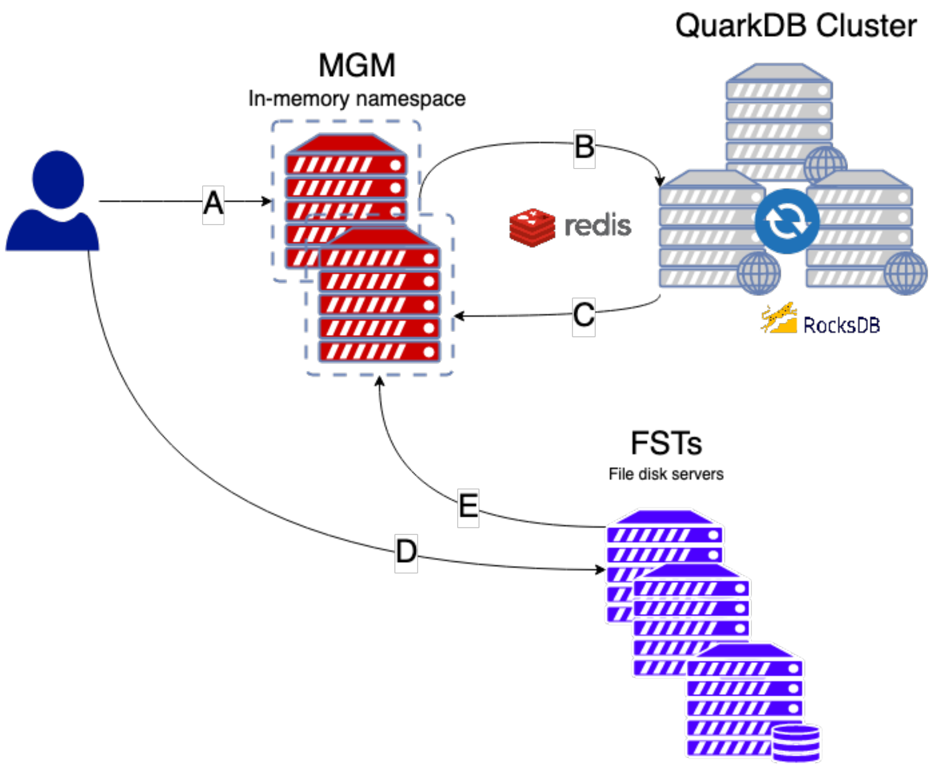
EOS, the LHC Disk Storage

EOS is used as the storage for the CERNBox service. It provides a replicated data storage with different ways to access the data: through a FUSE mount, XROOTD [8] protocol and WebDAV protocol. To ensure consistency across synchronization clients, from the web access and from FUSE or XROOTD clients, EOS storage extends the standard WebDAV protocol to include the synchronization semantics used by ownCloud, therefore a change made using the EOS command line interface is synchronized automatically to an end user computer through ownCloud synchronization clients. In the (Figure 2) the EOS Architecture is shown.

User requests are managed by the EOS manager node (named *mgm*) which are redirected to the source or destination file server (fst) depending of the type of request. The name space information is stored in a 3 node cluster using *RocksDB* variant developed at CERN, codenamed *QuarkDB* [9].

EOS provides a FUSE interface, so users can mount a particular remote directory in EOS as a local folder in their workstation. The fact of having remote resources in a local folder allows the use of tools for data analysis that only work in local file-systems. As a consequence, physicists could use analysis frameworks like ROOT to analyze the raw data.

Figure 2. EOS Architecture



This is one of the innovative use cases that CERNBox provides to the scientific community, the ability to interact with high valuable data in a non-disruptive and automatic way.

2.2 Legacy backup system limitations

CERNBox legacy backup is an in-house developed, event-based system. The backup is done by file and there is no concept of snapshots. Therefore, restoring a folder as it was in a certain moment is a complex task. The process of restoring files is triggered via support ticket and performed by the service operators. The backup repository is stored using the same technology (EOS) and in the same physical data center as the source data. This is not ideal as both source data and backup data could be compromised during a natural or software disaster scenarios.

2.3 CERNBox tree structure

CERNBox storage is organized in 8 different EOS physical clusters, and the data organization is represented in the following structure:

```
eos/
└─home-a/(physical instance)
   └─agustin/
```

```
|
|   |
|   |__alberto/
|__home-b/ (physical instance)
|__home-r/ (physical instance)
|   |__rvalverd/
|__project-a/(physical instance)
    |__atestproject/
```

Users data is stored in different physical instances depending on the starting letter of the username or project name. This structure gives flexibility in terms of backup organisation. The backup of home and project folders can be treated in an isolated way. Therefore, the backups can run concurrently as they can live in separate backup sets. This is clearly an advantage compared to the approach of backing up at the starting-letter level.

3 New backup solution: *cback* [10]

3.1 restic: the foundation

restic [11] is an open source technology to perform backups. It provides a simple command line interface that opens the door to an advanced backup functionalities. It diverges from the traditional backup systems that tend to be heavy, complex and very opaque. The tool is operated via a few commands and offers all the features expected by a modern backup system: de-duplication, encryption, backup verification, backup purging and restore flexibility. Similarly to git, it uses a repository structure to save the snapshot based backups. One of the key features of *restic* is called *Content Defined Chunking* [15]. This characteristic is especially noticeable when backing up small modifications of big files because only the chunks with the modifications are uploaded. This functionality allows an efficient usage of the available bandwidth. *restic* is also compatible with several storage back-ends, including S3 based back-ends.

3.2 Overview

cback is a highly scalable orchestration system for *restic* developed at CERN that provides all the tooling needed to manage the whole life-cycle of backup data: from backup and restore, to verification and retention policy management. Although *cback* is optimized to be used as a CERNBox backup, it can be used to backup arbitrary local or network file systems. The system is composed by stateless agents that perform all the different tasks provided by the system: backup, restore, prune, and verify. All the agents are coordinated through a central database which contains the job definitions.

Unlike normal backup systems, *cback* is meant to run as an always-running backup system, which means that the backup agents are looking for new jobs to process continuously in a configured interval. The main benefit of this approach is that it helps to distribute the backup activity in time and avoids user visible impact.

The main component of *cback* is the *cback backup* agent, which is responsible of executing the backup jobs. The restore jobs are handled by the *cback restore* and analogously *cback prune* performs the pruning/purging based on the configured retention policy. Additionally, the *cback verify* agent is responsible of performing the backup verification jobs. The *cback switch* component is responsible of changing the status of the backup jobs that need to be performed again. This decisions are based on job expiration parameters that indicates the validity of a successfully completed job.

3.3 Structure of the backup repository and caching policy

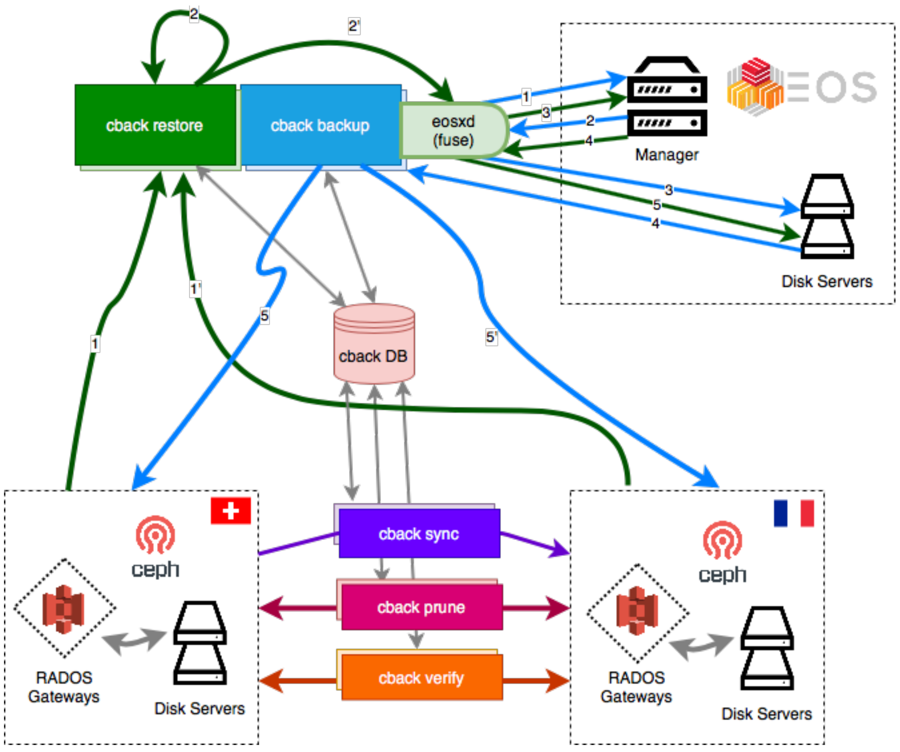
In *cback*, every backup jobs has a different backup repository. This granularity helps in keeping the backup repositories smaller and easy to operate. Thanks to this isolation between the different repositories, potential issues affecting one repository only affect the data contained in that particular repository. With this layout, applying data life cycle policies to the user backup repositories is a trivial task. Another benefit of this setup is that allows to perform a priority-based restore in the aftermath of a disaster event, which would allow restoring critical data in the first place required for business continuity.

restic uses by default a local cache to store a copy of the repository indexes and meta data in order to speed-up the backup process while looking for changes in the source folder. The size of this cache can vary from a couple of megabytes to even gigabytes depending of the backup data volume. The nodes where the agents run have a limited local storage and it was not feasible to preserve a copy of the cache for all repositories. Investigating this issue further, we found that the impact of the cache was specially noticeable in long backup jobs, and in this type of jobs, the time to re-create the cache was minimal compared with the time to perform the backup. Therefore, the decision of removing the cache after each backup was taken in order to preserve the stateless status of the backup agents and to minimize the requirements on local storage.

3.4 Data flow diagram

Figure 3 identifies different *cback* components and services and illustrates the process of data movement.

Figure 3. Data flow



As shown in Figure 3, the backup (blue arrows) reads the data from the EOS fuse mount, which communicates with the EOS manager to get the physical node and location of the requested file. Depending on the job configuration, *cback* transfers the data to the suitable S3 endpoint. For restore jobs (dark green arrows), the process is the opposite, but in this case, depending on the restore destination parameter, the data can be placed locally on the restore agent or remotely in EOS using again the fuse mount. The other *cback* agents work directly against the S3 endpoints, without any interaction with EOS. In all cases, the centralized *cback* database is used for coordination.

3.5 Backup Optimizations

Backup only when needed

Thanks to some optimization of the CERNBox source storage back-end (EOS), *cback* can decide if a backup is required or should be skipped. EOS provides a recursive *ctime* attribute, which is compared with the time stamp of the last backup stored. If this value is not newer, the backup is not performed at all. This avoids traversing the file system there are no changes. This feature is also supported in source file systems based on Ceph [12] fuse implementation (*CephFS*) thanks to the extended recursive attributes.

Avoid file opens

restic performs a *fopen* system call to open files when detecting changes. The file opening over a network file system like EOS means opening the file directly on an remote disk. This remote opening increases the latency of the detection phase of the backup because each file is opened for a short period. This round-trip per file is non negligible. We have observed that millions of files were opened for a very short period of time before being closed when running the restic backup. The code path in restic that was responsible for this behaviour was identified and improved. This optimizations has been merged in the upstream project[17] for the benefit of the restic community.

The proposed optimization drastically reduces the backup discovery time (between 30% and 50%) on network file systems (including EOS).

3.6 Metrics and service insights

cback is able to send metrics about the result of the different processes to Graphite [13] based endpoints. These metrics include the result of the process, and counters regarding the number of files or size of the data added to the backup repositories. Figure 4 shows the variability in terms of file/directories been considered per backup job.

Figure 5 depicts the duration of all jobs executed in 7 days which confirms that all backup jobs are performed in less that 24h (red marker).

4 Architecture

The architecture of *cback* is simple, just a database and an job executor and management agent. The database acts as control plane to set job definitions and track status. The agent executes and manage all the defined jobs and takes different role depending of the task to perform. Most of the type of agents act as a wrapper to *restic* internal commands. Every agent can run in same or different nodes and can be scaled as needed (no persistent data is stored locally). *cback* agents have the possibility of running the job in a interactive way or in a unattended and continuous mode.

Figure 4. Data added/deleted in 24h, 1h-summarized

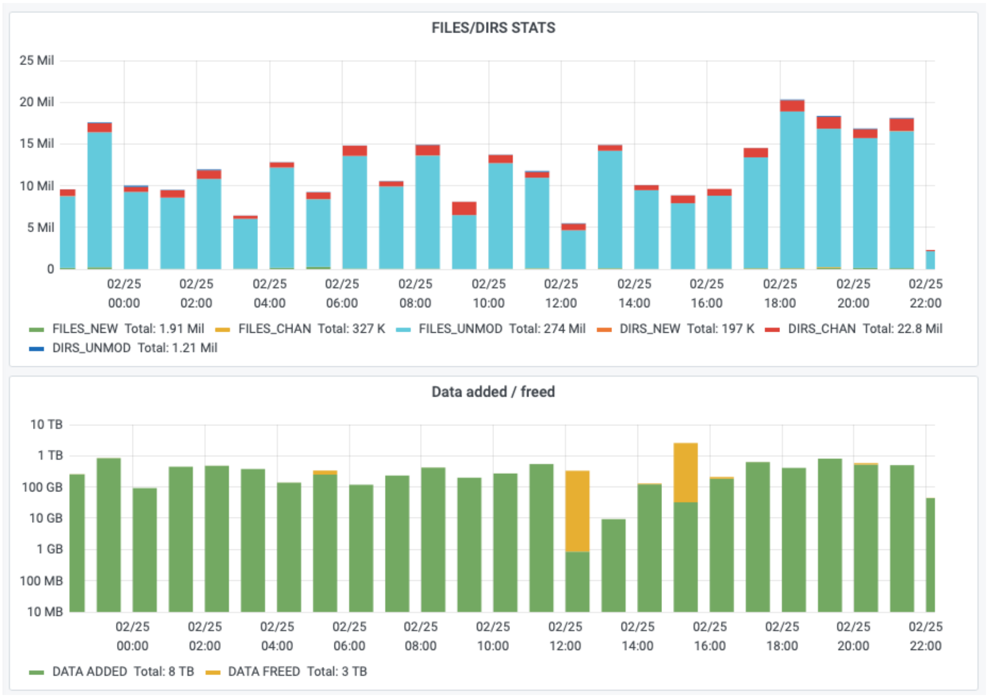
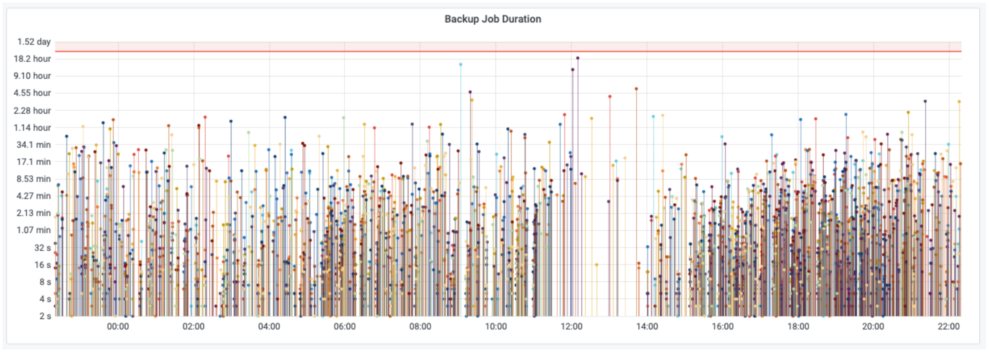


Figure 5. Backup job duration (7d)



In figure 7, a diagram of the architecture specifically for the deployment of CERNBox backup is shown. Stateless agents (center) are coordinated via a centralised database and they select jobs based on a *pulling* method. Each backup job stores the data on a different repository, which corresponds to a different S3 bucket.

4.1 Operation

For the CERNBox backup system, all the *cback* components are running in agent mode, which means that they are running continuously working in an unattended way in the background.

Figure 6. *cback* summary dashboard

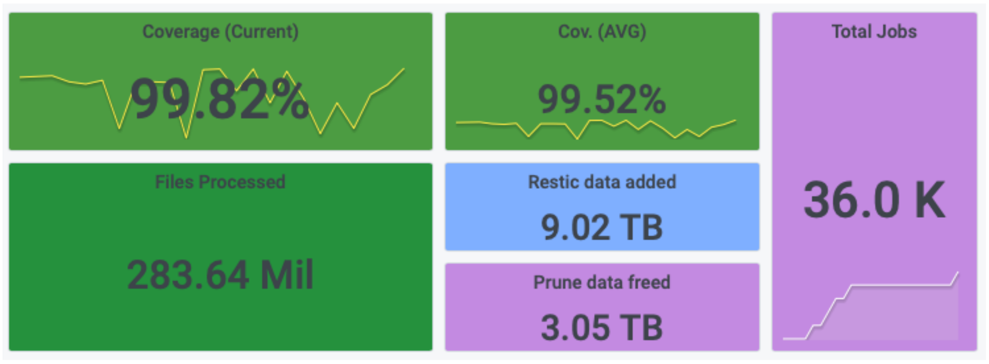
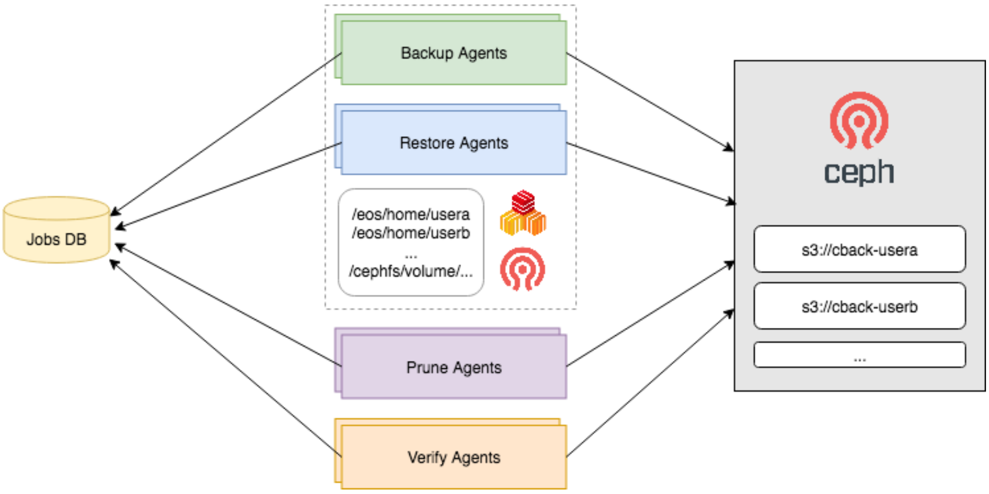


Figure 7. *cback* Architecture

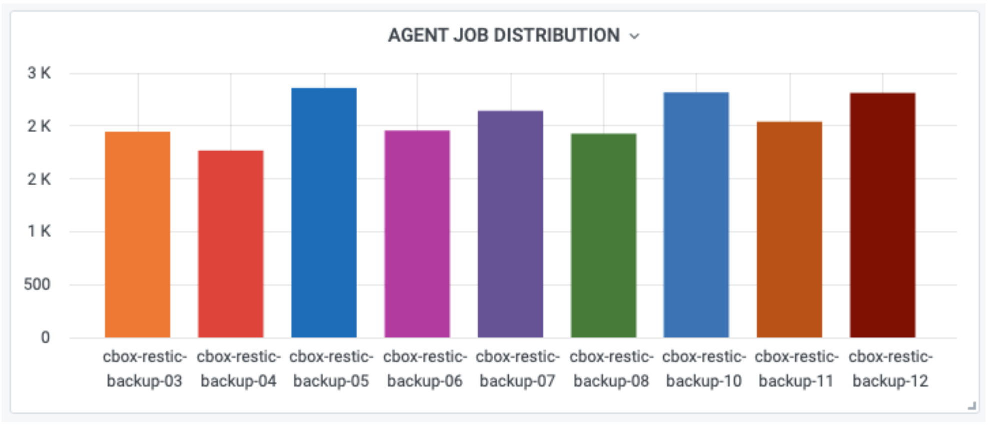


The backup agents query every 60 seconds the database for jobs to be in *Pending* status. If they exists, one job is selected based on a configured policy (random, older). The *cback backup* agent extracts the job definition, and performs the backup from the source on EOS fuse view to the specified destination. The appropriate S3 configuration is loaded dynamically from disk depending of the specified S3 endpoint. Once the backup is finished, the *cback backup* agent updates the database with information of the new status of the backup. At the end of the process, a set of metrics is sent to CERN's *Graphite* instances and presented in *Grafana* [14] dashboards.

Every hour, the *cback prune* agent queries the database for jobs in backup *Completed* status and prune status in *Pending*. *cback prune* selects a random job and applies the configured retention policy on the backup repositories in S3 and removes old snapshots if needed. 7 daily snapshots, 5 weekly snapshots and 6 monthly snapshots are kept for CERNBox. The *cback verify* agent works the same way, but querying for jobs in backup status *Completed*, prune status *Completed* and verify status in *Pending*.

The *cback switch* agent is the responsible of decide when a backup, prune or verify job is outdated and needs to be performed again. For the CERNBox use case, a backup is considered outdated after 24 hours of its completion, a prune 72 hours after, and a verification job after 30 days. The start of the next backup depends on the end time of the last one. The advantage of this approach is that the execution of the backup jobs is naturally scattered in time which helps to do not stress the storage back-end.

Figure 8. Backup job distribution (24h)



On Figure 8 can be observed that this model, although is simple, offers a good spread of the jobs across the different agents. On the other hand, due to this organic way of executing the jobs it would be difficult to have an overview of the global backup job status at any given time. For that, the concepts of 'backup coverage', 'prune coverage' and 'verify coverage' are used in *cback* command line tool. A backup coverage is a percentage that indicates the percent of backup jobs which were completed successfully in the last 24 hours (or within the configured backup interval). This value is computed by *cback* status every time is executed, giving to the administrator a quick and easy way to, through one simple value, know the overall status all of the backup jobs at any given moment. The same principle is used for the prune and verify coverage (check Fig. 9).

4.2 Components

The following section covers the definition and description of the data and storage management for *cback*

4.2.1 Database

cback uses a centralized MySQL/MariaDB database in order to store the current status information of the backup and restore jobs. The database design is optimized in order to reduce the overhead while doing basic operations. This optimization includes the aggregation of different entities on the same table in order to avoid an excessive use of JOINS for information retrieval. The current database model is illustrated in the Entity Relation Diagram presented in Figure 10.

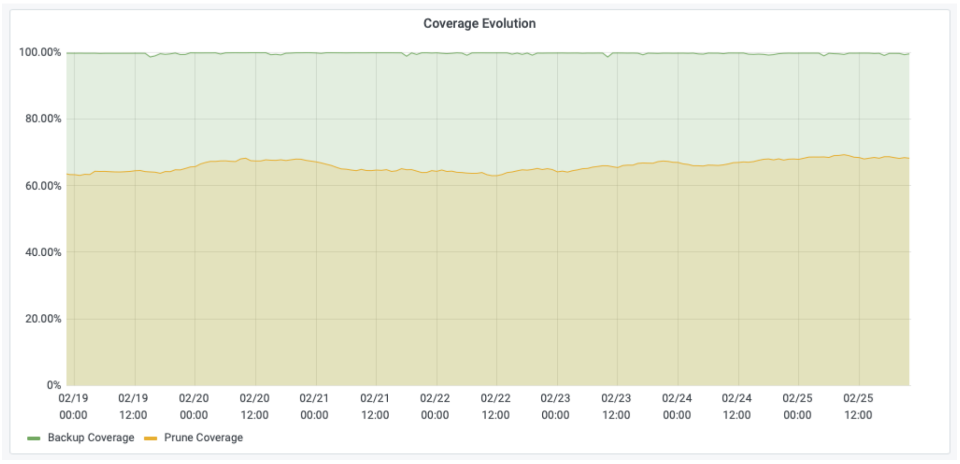
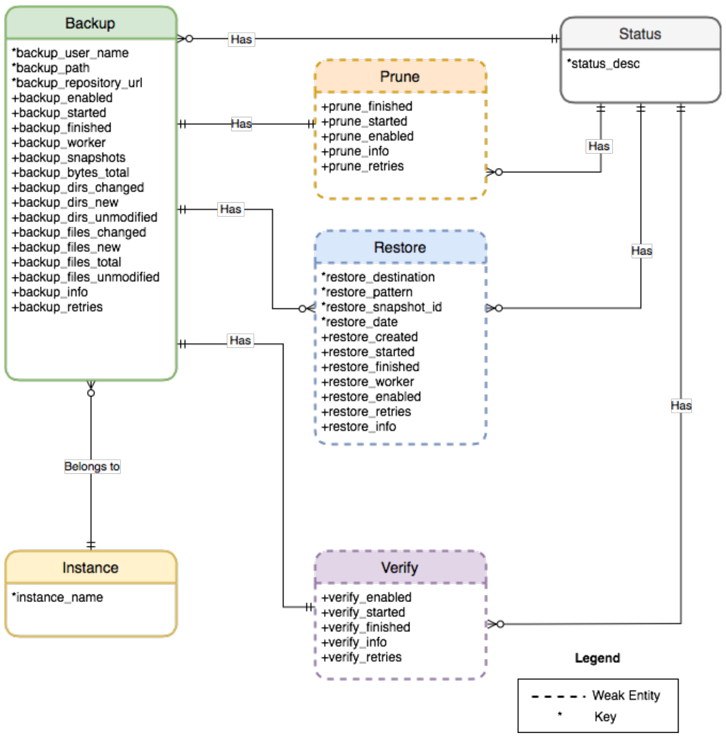


Figure 9. Backup and prune coverage over time (7 days)

Figure 10. Entity Relationship Diagram



4.2.2 Backup endpoints

Doing the backup of large volumes of information undoubtedly involves the need of a huge storage capacity.*restic* minimizes the the cost thanks to its de-duplication features. On the other hand, the backup data should always be available in order to support the restore of files. *cback* stores the backup in CERN S3 infrastructure, which is based on a Ceph implementation of AWS S3 using Rados Gateways. By default, Ceph uses replica 3 approach to store the objects which gives you an overhead of 200 percent in respect of the original data. Due to this, a configuration of erasure coding 4+2 is applied that gives an overhead of 50 percent while retaining the same availability. The availability of the Rados GW S3 endpoints is given by a dedicated load-balanced pool of nodes specifically for *cback* agents. As Ceph is configured to use a rack as a failure domain, the backup data would be accessible even if two racks are not available.

5 Further Work

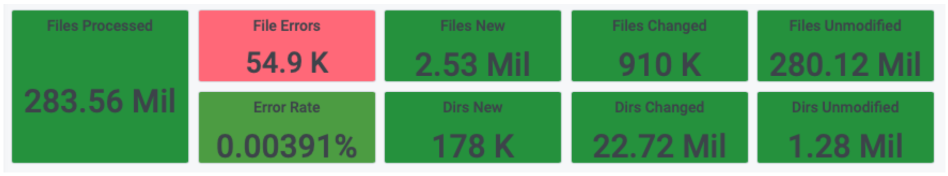
5.1 Avoid tree traversal

As stated before, *cback* implements a feature to skip backups when not needed. This helps to reduce the number of backup jobs that runs daily. On the other hand, in case of a single file modification, the full tree has to be traversed which is not efficient and produces unneeded load server side. As shown in Figure 11, vast majority of the files are unmodified during the daily backup jobs. This is clearly a field to improve. Different options can be explored to address this issue, like using *restic*'s *–files-from* functionality to specifically tell *restic* the exact set of files that changed (this information can be gathered from source file system in advance). Another solution can be to improve *restic* folder traverse logic to take benefit of recursive mtimes and being able to discard unchanged trees.

5.2 Restore "self-service"

The possibility of running asynchronous restore jobs opens the door to expose or offer the restore mechanism to the end users. The system could be interfaced to be used via CERNBox web UI or command line to submit restore jobs that would be processed automatically by *cback*. This would reduce the operational costs (as no action would be needed by the service manager) and improve the user experience.

Figure 11. File stats monitoring (24h)



Files Processed	File Errors	Files New	Files Changed	Files Unmodified
283.56 Mil	54.9 K	2.53 Mil	910 K	280.12 Mil
	Error Rate	**Dirs New**	**Dirs Changed**	**Dirs Unmodified**
	0.00391%	178 K	22.72 Mil	1.28 Mil

5.3 Multi-replica Backups and non-S3 storage backend support

Currently, *cback* can perform the backup in different S3 zones. Having two copies of the backup is already possible (defining two backup jobs) but the concept of producing a configurable number of exact backup replicas in different S3 geographical zones can be explored.

cback only supports S3 as the back-end for the backup repositories. *rclone* recently implemented a feature to serve *restic* repository from any configured back-end. This development would enable the support for different storage systems but limiting the usage of *rclone* as the unique *cback* back-end.

6 Summary

cback is a fully-featured system that allows the execution and management of the backup for big data sets.

With *cback*, the data to be backed-up is filtered to ensure that only important data is preserved. The *prunning* system and techniques like de-duplication and erasure code layouts allow to reduce the storage usage and cost.

Apart from the job definition database, all the different components of *cback* are stateless and can be scaled vertically or horizontally. The backup repositories are stored in a highly scalable distributed object storage system powered by Ceph. Data is also stored in another geographical location and using a different storage technologies, which is a safe mechanism against natural and software disasters.

Automatic backup verification ensures that data stored is consistent and can detect integrity issues. The backup data is transferred via secure HTTP, stored encrypted using *AES-256* and authenticated using *Poly1305-AES*.

cback restore can restore the files directly on the users folder that allows the possibility to offer a self-service restore mechanism.

7 Conclusions

A multi-petabyte backup solution based on *cback* has been introduced and is in full operation to protect CERNBox service agains disaster scenarios. More than 37,000 user accounts and 800 projects are backed up every day using *cback*.

cback complements *restic* backups by orchestrating its core functionalities to the scale needed by big data sciences. *cback* is a fully featured and modern backup system that helps protecting CERNBox users and project data every day. The presented solution is an example of how to address the backup of a multi-petabyte distributed data repository. The inner workings of this systems can be extended to other use cases that involve similar constraints (refer to section 2).

References

[1] H. G. Labrador et al, EPJ Web of Conferences **214**, 04038, *CERNBox: the CERN storage hub* (2019)

[2] H. G. Labrador, *CERNBox: Petabyte-Scale Cloud Synchronisation and Sharing Platform* (University of Vigo, Ourense, 2015) EI15/16-02

[3] Mascetti, Luca and Labrador, H Gonzalez and Lamanna, M and Mościcki, JT and Peters, AJ, Journal of Physics: Conference Series **664**, 062037, *CERNBox + EOS: end-user storage for science* (2015)

[4] Peters, AJ and Sindrilaru, EA and Adde, G, Journal of Physics: Conference Series **664**, 062037 (2015)

[5] Peters, Andreas J and Janyst, Lukasz Journal of Physics: Conference Series **331**, 052015 (2011).

[6] "CERNBox", "The cloud storage solution from CERN", https://cernbox.web.cern.ch

[7] "EOS", "EOS Open Storage", https://eos.web.cern.ch (access time: 12/02/2021)

[8] "XRootD", "The XROOTD project", https://xrootd.slac.stanford.edu (access time: 12/02/2021)

[9] Bitzes, Georgios and Sindrilaru, Elvin Alin and Joachim Peters, Andreas, EPJ Web of Conferences, Volume 214, id.04019 Scaling the EOS namespace - new developments, and performance optimizations

[10] R. V. Cameselle, *Distributed Backup for High Volume File Systems* (University of Vigo, Ourense, 2020) MEI 19/20-001
[11] "restic", "Backups done right!", https://restic.net (access time: 12/02/2021)
[12] "Ceph", "The Future of Storage", https://ceph.io (access time: 12/02/2021)
[13] "Graphite", "Make it easy to store and graph metrics", https://graphiteapp.org (access time: 12/02/2021)
[14] "Grafana Labs", "Explore metrics and log", https://grafana.com (access time: 12/02/2021)
[15] "restic Community", "Introducing Content Defined Chunking (CDC)", https://restic.net/blog/2015-09-12/restic-foundation1-cdc (access time: 12/02/2021)
[16] "rclone", "Rclone - rsync for cloud storage", https://rclone.org/ (access time: 12/02/2021)
[17] "restic file open patch", https://github.com/restic/restic/pull/2970 (access time: 12/02/2021)