# *A Tape RSE for Extremely Large Data Collection Backups*

*Andrew* Hanushevsky[1*], *Guangwei* Che[1], *Lance* Nakata[1], and *Wei* Yang[1]

[1]SLAC National Accelerator Laboratory, 2575 Sand Hill Rd., Menlo Park, CA 94025, USA

**Abstract.** The Vera Rubin Observatory is a very ambitious project. Using the world's largest ground-based telescope, it will take two panoramic sweeps of the visible sky every three nights using a 3.2 Giga-pixel camera. The observation products will generate 15 PB of new data each year for 10 years. Accounting for reprocessing and related data products the total amount of critical data will reach several hundred PB. Because the camera consists of 4kx4k CCD panels, the majority of the data products will consist of relatively small files in the low megabyte range, impacting data transfer performance. Yet, all of this data needs to be backed up in offline storage and still be easily retrievable not only for groups of files but also for individual files. This paper describes how we are building a Rucio-centric specialized Tape Remote Storage Element (TRSE) that automatically creates a copy of a Rucio dataset as a single indexed file avoiding transferring many small files. This not only allows high-speed transfer of the data to tape for backup and dataset restoral, but also simple retrieval of individual dataset members in order to restore lost files. We describe the design and implementation of the TRSE and how it relates to current data management practices. We also present performance characteristics that make backups of extremely large scale data collections practical.

## 1 Introduction

The Vera Rubin Observatory [1], completing construction in Chile, is a trailblazing astronomical facility that plans to revolutionize the study of the universe. The main mission, the Legacy Survey of Space and Time (LSST) [2], will use the Simonyi Survey Telescope [3], featuring an 8.4-meter mirror with a 3.2 gigapixel camera, to capture an unprecedented wide and deep image of the night sky every few nights. The deep survey allows detecting changes in the universe across time facilitating the study of dark matter and energy, real-time monitoring of transient events, identifying potentially hazardous asteroids, and producing the most detailed map of the Milky Way Galaxy as well as the near universe for future scientific breakthroughs.

These observations will generate 20 terabytes of images per night, plus derived data products such as catalogues. At the end of the 10 year survey, approximately 200 petabytes of raw data and 300 petabytes of derived data products will exist across billions of files.. All of this data needs to be backed up to recover in the event of a disaster. This paper describes the system we have devised to make a backup and data recovery possible at scale.

* Corresponding author: abh@slac.stanford.edu

### 1.1 Basic data management components

The main Rubin data center is located at SLAC and named the USDF (US Data Facility) where data is first ingested, some of which is locally processed while the rest is sent to other data centers (i.e. France's FRDF and the United Kingdom's UKDF) for processing and whose output is sent back to the USDF [4,5]. In order to provide a cohesive workflow, all of the data is registered in a data management system named Rucio [6]. The data management system is responsible for farming out the data and keeping track of all data products.

In order to provide efficient access to the data, the USDF employs a high performance file system named Weka [7]. It is a POSIX file system backed by an object store that is fully distributed and makes all of the data accessible to any system that mounts the file system. Hence, all data products registered in Rucio are accessible via the Weka file system.

The final key data management component is IBM's tape-based High Performance Storage System (HPSS) [8]. The storage system uses tape to store data and makes it accessible either for storage or retrieval via pftp [9], HSI [10], HTAR [11], as well as a FUSE [12] mounted file system. The HPSS component is meant for data archival and backup.

The three basic data management components: Rucio, Weka, and HPSS provide the infrastructure needed for a fully functional backup and recovery system, and that system simply needs to co-ordinate the use of all three foundational components.

## 2 The Backup System

As noted in the previous section, the goal is to use the existing infrastructure components to create a backup and recovery system. This simply requires that the system co-ordinate the use of the components to copy un-backed up data to an archive and facilitate its retrieval when needed. Our approach was to use the well known XRootD [13] framework to create a service that does just that. In fact, the XRootD framework already provides the basic modules to interact with the existing infrastructure. All that is needed is to properly sequence the data flow: 1) identify what needs to be backed up, 2) create a backup data unit, and 3) archive the unit on permanent storage (i.e. tape).

### 2.1 Backup Architecture

#### *2.1.1 Identifying what needs to be backed up*

Identifying what needs to be backed up is best left to the data administrator. We decided to use Rucio meta-data features for this purpose as they integrate in how data is managed in the Vera Rubin Observatory. Collections of related data are grouped in dataset units. These units are processed, sent to other data facilities, which, in turn, create new datasets. The obvious observation here is that the natural backup unit is a dataset. Once created it may need to be backed up. We added a Rucio metadata variable, arcBackup, to each dataset. Its initial value is *none*. Should the dataset need to be backed up, the data administrator sets the value to *need*. When the dataset is actually backed up, the value is set to *done*. The metadata approach has several upsides in that a) it provides an easy was to audit what needs to be backed up and what has been backed up, and b) guarantees that a dataset will eventually be backed up as its status remains in "*need*" until ait is actually backed up. The downside is that the whole Rucio dataset catalogue needs to be scanned to determine the

backup requirements. However, given that backup processing is relatively slow-paced, scanning requirements are minimal (e.g. a couple of times a day).

*2.1.2 Creating the backup unit*

In the previous section we noted that a backup unit is a dataset. However, datasets are made on hundreds of individual files and these files vary in size from several hundred bytes to several gigabytes. Creating a physical backup using individual files is not sustainable as tape systems require large transfer units to maintain transfer speed and minimize tape wastage. Ideally, each backup of a collection of files should be represented as a single file. In order to accomplish this, we package all dataset files into a single zip archive file. The Weka file system made this task relatively easy. All files registered in Rucio are accessible via the Weka file system. All we needed to do is create a directory structure of symlinks to the individual dataset files. That directory structure represented the complete dataset. We could then invoke the zip command to package the directory contents into a single zip archive. That archive became the backup unit and is in 1-to-1 correspondence with what is registered in Rucio. The path of the archive corresponds to the dataset name converted to a path. The actual filename of the archive is simply Archive.zip.

The upsides are that a) the backup unit is large enough to maintain tape efficiency, and b) individual files are easily accessible for retrieval when needed. The downside is that the backup unit may become too large, complicating the creation of the zip archive. However, based on the LSST data processing needs; it appeared unlikely that a backup unit would ever become excessively large.

*2.1.3 Archiving*

This turned out to be the easiest part. Since the HPSS can be FUSE mounted, the act of archiving simply means moving the zip archive created in the previous section to a mounted HPSS directory. From there, HPSS makes sure that the data is copied to tape.

**2.2 Implementation**

The implementation closely follows the infrastructure pieces being used (i.e. Rucio, Weka, and HPSS). Figure 1 shows how the pieces fit together.

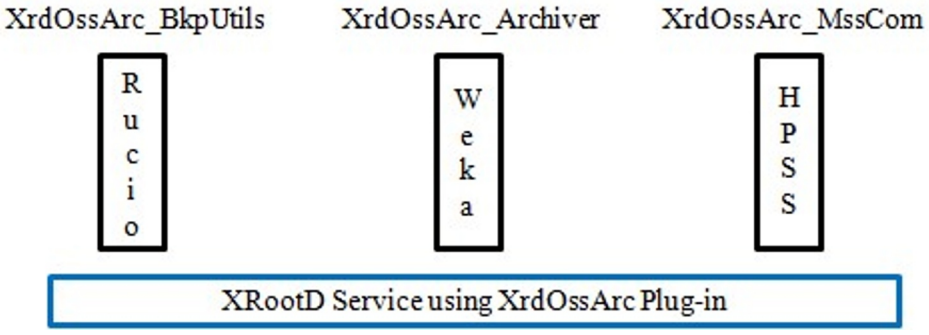


**Figure 1Archiving Implémentation Architecture**

The XRootD service, implemented as a plug-in (i.e. XrdOssArc) to an XRootD server, coordinates and drives the three vertical columns. Each column is actually a Python script:

**XrdOssArc_BkpUtils** handles all functions related to Rucio. These include listing datasets that need to be backed up and manipulating metadata variables.

**XrdOssArc_Archiver** handles all function that relate to the Weka file system. This includes creating symlinks to the files contained in the dataset, generating the zip file archive, and moving it to the HPSS mounted directory for archiving.

**XrdOssArc_MssCom** is used to communicate with HPSS. This includes queries of whether an archive file is on tape or in the disk buffer.

While the XRootD plug-in is a C++ program, we chose to write the specific infrastructure handlers in Python. The choice was largely driven by the fact that the Rucio API is written Python as well. Additionally, as the system needs to evolve due to infrastructure changes, it is far easier to modify and redeploy a Python script than C++ components. Finally, other sites could customize the system to correspond to their infrastructure components by replacing the appropriate scripts.

## 3 The Recovery System

The dataset and file recovery is fairly straightforward. To recover a full dataset one can copy the Archive.zip file for the dataset to the destination, unzip the contents and move the files into the proper locations. The path to the backup is shown below.

/backup/*dataset_DID*~Archive.zip

Unfortunately currently there are no tools that automate the dataset restoration process, which is a time consuming effort. Should the Archived dataset be registered in Rucio as being located in the tape RSE (i.e. the XRootD service that created the backup) then Rucio could replicate the dataset to the RSE that no longer has it. However, this would be a slow process as copying hundreds, if not thousands of files, many which are small, across the network is hardly efficient.

Recovering a small set of individual files from a dataset, the common case, simply requires copying the files from the XRootD service that created the backup using a specially formatted path, shown below.

/backup/*dataset_DID~file_DID*

### 3.1 Implementation

The XRootD framework includes an extensive file service set of plug-ins. The recovery service uses these plug-ins to access files in the HPSS disk buffer. The storage system plug-in is, however, wrapped with a specialized plug-in that handles access to the archive zip file and files within the archive zip file. This plug-in is named XrdOssArc and is the same plug-in that is used to create backup archives.

Whenever a client attempts to copy a file that starts with /backup, the XrdOssArc plug-in determines whether the client was the full archive or an individual file within the archive. In either case, the archive is brought into the HPSS disk buffer by simply opening the file as HPSS will automatically stage in the file if it is not already in the disk buffer. If the complete archive is wanted, it is served directly from the disk buffer. If the client wants a single file from the archive, the XrdOssArc plug-in provides access to the archive member as if it were an actual stand-alone file. Thus, the file need not be extracted or copied elsewhere. This, however, requires that the archive is not compressed.

## 4 Future Work

While creating backups is architecturally complicated, the client interface is relatively simple. Hence, backups can occur with minimal human intervention. On the other hand, recovery is architecturally simple but, unfortunately, requires significant human intervention. We are studying what changes can be made to minimize the recovery effort. Fortunately, recovering individual files is fairly straightforward and this is the main Rubin Observatory requirement. Disaster recovery is, well, a disaster in any case and would likely require significant effort no matter the system design.

Not all of LSST data is registered in Rucio and we anticipate that at least some of that data will need to be backed up. We plan to follow up on this scenario and devise a plan on how such data would architecturally fit into our backup model.

Our next version of the RSE backup system will automatically break up a dataset into multiple archives, each using a configurable optimum size, as needed. We have also, devised an algorithm that can identify which dataset file is in which archive without the need of recording a full manifest. This makes it possible to recover individual files by restoring only the particular archive they are in.

As of this writing, the backup and recovery systems have not yet been deployed for production use. However, that have been extensively tested and found to work well. We look forward to seeing if our approach works at scale.

## 5 Conclusion

The Rucio Tape RSE backup system works well in a Rucio type of environment. The ability to tailor the system with replaceable components also makes it not only possible to extend to other data management schemes but also address its shortcomings in a production environment without drastic changes. We have found that using Python for key interactions with existing infrastructure significantly lessened the time to effect needed changes. This is especially true for Rucio which comes with a robust Python API. We have also found that we cannot depend on all datasets to be roughly a size that is optimal for tape storage. Indeed, some will likely be substantially larger, hence we plan to update the system to handle archive splitting as noted in the previous section.

## Acknowledgements

This material is based upon work supported in part by the National Science Foundation through Cooperative Agreement AST-1258333 and Cooperative Support Agreement AST-1202910 managed by the Association of Universities for Research in Astronomy (AURA), and the Department of Energy under Contract No. DE-AC02-76SF00515 with the SLAC National Accelerator Laboratory managed by Stanford University. Additional Rubin Observatory funding comes from private donations, grants to universities, and in-kind support from LSSTC Institutional Members.

This work has been supported by the UK Science and Technology Facilities Council (STFC) funding for UK participation in LSST, through grants ST/X001334/1 and ST/Y003004/1.

## References

[1] Vera C. Rubin Observatory, "Vera C. Rubin Observatory," [Online]. Available: https://rubinobservatory.org/. [Accessed 25 January 2025].

[2] Ivezic, Zeljko, et al., "LSST: From Science Drivers to Reference Design and Anticipated Data Products," *The Astrophysical Journal,* vol. 873, no. 2, p. 44, 2019, DOI: 0.3847/1538-4357/ab042c.

[3] Thomas, Sandrine J, et al, "Rubin Observatory Simonyi Survey Telescope status overview," *Proceedings of the SPIE,* vol. 12182, p. 18, August 2022.

[4] Hernandez, Fabio, et al., "Data Movement Model for the Vera C. Rubin Observatory," in *Proceedings: 27th International Conference on Computing in High Energy and Nuclear Physics (CHEP 2024)*, Krakó, Poland, 2024.

[5] Yang, Wei, et al., "Preparation for Multi-Site Processing at the Vera C. Rubin Observatory," in *Proceedings: 27th International Conference on Computing in High Energy and Nuclear Physics (CHEP 2024)*, Kraków, Poland, 2024.

[6] Barisits, Martin, et al., "Rucio: Scientific Data Management," *Computing and Software for Big Science,* vol. 3, p. 11, 2019, DOI: 10.1007/s41781-019-0026-**3.**

[7] E. Frank, M. A. Hall and I. H. Witten, "The WEKA Data Mining Software: An Update," *SIGKDD Explorations,* vol. 11, no. 1, 2016, DOI: 10.1145/1656274.1656278.

[8] HPSS Collaboration, "High-Performance Storage System (HPSS," [Online]. Available: https://www.hpss-collaboration.org/. [Accessed 26 January 2025].

[9] IBM, "PFTP: syntax and options," [Online]. Available: https://www.ibm.com/docs/en/iis/11.7?topic=operator-pftp-syntax-options. [Accessed 26 January 2025].

[10] LLNL, "Using HSI: Hierarchical Storage Interface," [Online]. Available: https://hpc.llnl.gov/documentation/user-guides/using-hsi-hierarchical-storage-interface. [Accessed 26 January 2025].

[11] LLNL, "Using HTAR: HPSS Tape Archiver," [Online]. Available: https://hpc.llnl.gov/documentation/user-guides/using-htar-hpss-tape-archiver. [Accessed 26 January 2025].

[12] "FUSE: Filesystem in Userspace," [Online]. Available: https://github.com/libfuse/libfuse. [Accessed 26 January 2025].

[13] XRootD Collaboration, "XRootD: High Performance, Scalable Fault Tolerant Access to Data," [Online]. Available: http://xrootd.org/. [Accessed 26 January 2025].