

D9.5: Final Report on the Applicability of Object Stores within EUDAT

Author(s)	Maciej Brzeźniak (PSNC)
Status	Final
Version	v1.0
Date	06/12/2017

Document identifier: EUDAT2020-DEL-WP9-D9.5	
Deliverable lead	PSNC
Related work package	WP9
Author(s)	Maciej Brzeźniak (PSNC)
Contributor(s)	Benedikt von St. Vieth (FZJ), Stanisław Jankowski (PSNC), Ian Collier (STFC)
Due date	28/02/2017
Actual submission date	06/12/2017
Reviewed by	Genet Edmondson
Approved by	PMO
Dissemination level	PUBLIC
Website	www.eudat.eu
Call	H2020-EINFRA-2014-2
Project Number	654065
Start date of Project	01/03/2015
Duration	36 months
License	Creative Commons CC-BY 4.0
Keywords	Object stores, B2SAFE, scalability, reliability, performance

Copyright notice: This work is licensed under the Creative Commons CC-BY 4.0 licence. To view a copy of this licence, visit <https://creativecommons.org/licenses/by/4.0>. 

Disclaimer: The content of the document herein is the sole responsibility of the publishers and it does not necessarily represent the views expressed by the European Commission or its services.

While the information contained in the document is believed to be accurate, the author(s) or any other participant in the EUDAT Consortium make no warranty of any kind with regard to this material including, but not limited to the implied warranties of merchantability and fitness for a particular purpose.

Neither the EUDAT Consortium nor any of its members, their officers, employees or agents shall be responsible or liable in negligence or otherwise howsoever in respect of any inaccuracy or omission herein.

Without derogating from the generality of the foregoing neither the EUDAT Consortium nor any of its members, their officers, employees or agents shall be liable for any direct or indirect or consequential loss or damage caused by or arising from any information advice or inaccuracy or omission herein.

CONTENTS

EXECUTIVE SUMMARY	4
1. INTRODUCTION	5
2. ADVANCES IN OBJECT STORAGE TECHNOLOGY	7
2.1. Volume Reduction Technology	7
2.2. Interfaces and Standards	9
2.2.1. Standard Interfaces and their Developments	9
2.3. Object Storage Integration Options	10
2.3.1. Filesystem-like Interfaces to Object Stores	10
2.3.2. Native or Standard Interfaces-based Integration	14
2.4. Market Trends Related to Object Storage Integration	15
2.4.1. Long-term and Large Scale Storage	15
2.4.2. Long-term Storage and Large-scale Storage Management	15
2.4.3. Data Repository Software	18
2.4.4. Sync and Share Service	19
3. DISCUSSIONS AND FEEDBACK ON THE CONCEPT	21
3.1. Discussions within EUDAT	21
3.1.1. Basic Integration Concept	21
3.1.2. Consultancy within WP9	23
3.1.3. TC Feedback	24
3.1.4. Feedback from Service Activities	24
3.2. Feedback from Outside the Consortium	28
3.2.1. EC Reviewers	28
3.2.2. External Data Management and Cloud Services Experts	29
3.3. Interactions with User Communities	30
3.3.1. Europeana	30
3.3.2. Fusion	31
3.4. Summary of Discussion Outcomes	32
4. EXTENDED PROOF OF CONCEPT	33
4.1. Prototype Design	33
4.2. Implementation and Evaluation	34
4.3. Conclusions from the Proof of Concept	35
5. FROM OBJECT STORES INTEGRATION TO FUTURE ARCHITECTURE OF EUDAT	36
5.1. Scene for Future Architecture of EUDAT	36
5.2. Recommendations for Future Architecture of EUDAT	37
5.3. Summary	39
5.4. Possible Further Work	39

LIST OF FIGURES

Figure 1: Integration of object stores at the iRODS back-end	22
Figure 2: Object storage-based B2SAFE implementation	22
Figure 3: Envisioned future architecture of EUDAT CDI	38

EXECUTIVE SUMMARY

In this deliverable we provide a summary of the work done by Task 9.1.1 of EUDAT's Work Package 9 (WP9) which, in accordance with the project's description of work (DoW), examined the applicability of object stores for implementing the EUDAT Collaborative Data Infrastructure (CDI) and assessed the potential of this technology to improve CDI architecture components or to replace their current implementations.

This task was a part of WP9's activities that looked into modern architectures and technologies in order to examine and evaluate their usability for providing the basis for low-maintenance-cost, functionality-rich and high-quality EUDAT services for EUDAT's user communities. In this wider context, T9.1.1 analysed the potential impact of applying object stores to the architecture and implementation of particular EUDAT services (namely B2SAFE, B2SHARE, and B2DROP) and also provided a view on possible architecture developments for the EUDAT CDI in the future.

T9.1.1 was a task spanning 24 months. In this M24 deliverable we report the final conclusions of that work, based on activities conducted in the 2nd year of this phase of the EUDAT project, including further analysis of the technology, and proofs of concept with regard to selected areas of the application of object stores, as well as discussions of the results, concepts and ideas worked out during the first year of the work on this task. The consultations that were undertaken as part of this task were conducted within WP9, and within EUDAT as a whole, as well as beyond the project consortium. In this report we also provide projections and proposals for the inclusion of our results in the technical development roadmap of EUDAT CDI. We envision a fully modular and scalable CDI architecture that includes objects stores as the reliable and scalable data persistency and access layer.

The work that was done in our second year confirmed that object stores have the potential to be used to implement a scalable, reliable, open, flexible and cost-effective, long-term sustainable data persistency layer that could be used as a basic building block for EUDAT services. The work also showed that object stores architectures, interfaces and available implementations make it possible to integrate them with both the current iRODS-based and future web-technology-based software stacks of the EUDAT CDI, although much of this functionality could be supplied by the object store itself with a reduced need for additional middleware components.

In particular, our proof of concept work demonstrated that B2SAFE (which is EUDAT's reliable, long-term storage service for registered data objects) can be efficiently and reliably implemented based on object stores by exploiting their internal mechanisms for data replication, as well as their interfaces for integrating external functionality such as persistent identifiers (PIDs). Our analysis also showed that there is potential for implementing and sharing an object storage-based back-end for other EUDAT services, including B2SHARE and B2DROP (assuming the relevant adaptations of these services).

1. INTRODUCTION

Task 9.1.1 "Object Store Evaluation" of EUDAT's Work Package 9 (WP9) was devoted to studying object storage concepts, as well as architectures and implementations, in order to assess their applicability to building scalable, interoperable, reliable and federated services for the EUDAT Collaborative Data Infrastructure (CDI). Task 9.1.1 was a subtask of T9.1 "HTTP-based Storage and Federations" aiming at exploring emerging technologies based on, and accessible through, HTTP. Task 9.1 was part of the activities of WP9, the technology exploration Joint Research Activity (JRA), that, in line with the description of activities (DoA), aims to enhance the EUDAT CDI by exploring new technologies that could improve or replace components within the EUDAT CDI.

The first year of work in this subtask focused on **studying the technology of object stores and proposing options for integrating object stores into the EUDAT CDI**. The former included study of the object storage concept and architectures, and the most popular open source implementations, as well as analysis of ghd features and functionalities of object stores in the context of the requirements of the EUDAT CDI. The latter encompassed a first round of proof of concept implementation of B2SAFE functionality based on OpenStack Swift¹ and its evaluation. We also proposed an approach for applying Ceph² as the base of B2SAFE, however, due to a more complex implementation, we have postponed the pilot for Ceph. While OpenStack Swift provides an elegant way of integrating additional functionality into the input/output (I/O) processing chain by using Python Paste³ framework, adding functionality to Ceph requires modifications of the core of Ceph which is not sustainable in long term. In the interim report from the twelfth month (M12) of the task we also gave an overview of the possible integration of object storage-based B2SAFE services with other layers of the EUDAT CDI.

In the second year of work, we continued the **analysis of the object storage technology**, including watching the trends related to object storage software and software-defined storage infrastructure. **We also continued research on the integration of object stores into the EUDAT services. While the main focus was B2SAFE**, we also analysed the possibility of integrating object stores at the back-end of B2SHARE and B2DROP. Details of this work and the main observations made during the second year of the project are presented in section 2.

We also conducted a series of discussions within WP9 (with the personnel involved in tasks 9.1.2 and 9.2 in particular) **and across various stakeholders in EUDAT** (including WP5) related to the possibility of integrating the results from task 9.1.1 into prototype or production versions of EUDAT services. Particular attention was paid to the integration and synchronization of the work and efforts with WP9's task 9.1.2 which is focused on HTTP-based federation. Based on these discussions, we prepared a vision of integrated object stores and federation components that make it possible to implement basic functionality for data storage and data management, as well as efficient, scalable and reliable data access based on a lightweight HTTP front-end. We also discussed the concept of integrating the envisioned object storage-based data persistency layer into the EUDAT services with relevant members of the EUDAT project, particularly those responsible for services such as B2SAFE, DPM (Data Policy Manager), B2SHARE and B2DROP. Details of these discussions and the main conclusions are presented in section 3.1.

We also **consulted stakeholders from outside of project**, including data management and cloud services experts at CERN, Geant, National Research and Education Networks (NRENs) and other R&D projects, as well as in industry. These consultations confirmed our beliefs about the applicability of object storage to large-scale, long-term scalable and reliable data management systems. Overall, there is a common understanding that the integration of object storage systems into data management projects, systems and services is both needed and profitable. This includes large-scale data management projects such as WLCG, and cloud

¹ <https://wiki.openstack.org/wiki/Swift>

² <http://ceph.com/ceph-storage/>

³ <https://pypi.python.org/pypi/Paste>

computing and storage services at NRENs and at universities. Details of these discussions are provided in section 3.2. That section also contains an analysis of the feedback from project reviews conducted by the EC.

In the second year of the project **we also presented the new opportunities and features of object storage-based services to the user communities** currently involved in EUDAT and to those considering usage of EUDAT CDI, including, amongst others, Europeana and the fusion community with the partners CCFE and ITER. Overall, these discussions revealed that the rapidly increasing needs of research communities regarding the volume of the data to be stored, preserved, accessed and made referencable, as well as the complexity of their data structures in terms of the number of objects, their relationships and the expected data deposition plus access rates (bandwidth and objects per second), require applying fully scalable, distributed and decentralized storage technologies. An additional effect of these interactions was the productions of proposals for pilot activities to be conducted in the integrated infrastructure project that will be follow-up to EUDAT, EGI and Indigo-DataCloud. Details of these discussions and the overview of future proposals are covered in section 3.3.

We also analyzed and discussed the applicability of the results from WP9's task T9.2 devoted to graph databases in the context of our work. We concluded that graph databases could be applied as a scalable metadata management solution that would be complementary to using object stores for data storage and access as well as basic data management. Importantly the potential of graph databases for representing even complex relationships between objects makes it possible to implement effective data exploration functionality in the integrated scalable data storage system.

In parallel to our analysis and discussions, we prepared a **proof of concept implementation** of the B2SAFE service solely based on object stores and graph databases. This prototype was based on the results from the first year, as well as on new ideas worked out in tasks T9.1.1 and T9.1.2 during the second year. OpenStack Swift was used as the data storage and access layer, and also implemented elements of data management (such as replication, integration of PIDs, and logging the namespace and object store content modifications), while the graph databases were applied for meta-data management, access and exploration. Details of this work are included in section 4.

In this wider context, T9.1.1 analysed the potential impact of applying object stores to the architecture and implementation of particular services (namely B2SAFE, B2SHARE, and B2DROP) and provided insight regarding the possible architecture development for the EUDAT CDI in the future. We also provided projections and proposals for the inclusion of our results in the technical development roadmap of the EUDAT CDI. Overall we envision a fully modular and scalable CDI architecture that includes objects stores as the reliable and scalable data persistency and access layer. The summary of this analysis is included in the final section of the document, section 5.

2. ADVANCES IN OBJECT STORAGE TECHNOLOGY

In the first year of the project, the work of task 9.1.1 focused on the analysis of object storage technology in the context of its general applicability for EUDAT services. We concluded that work with the belief that EUDAT could benefit from applying object stores to the CDI. This would make further improvements of the scalability, reliability and long-term sustainability of the services possible, thanks to increased performance, lower per-terabyte investment and operational cost of object stores, as well as greater openness of these systems. We also pointed that using object stores as a shared persistent data storage and access layer would improve the overall architecture by introducing modularity of the functional components, such as data storage vs data management and data access. It would also increase the adoption of industry standard application programming interfaces (APIs) and simplify the long-term development and maintenance of the system.

While new technology brings many promises, it is obvious that the integration of new APIs and the necessary re-design and adaptation of the services incurs important costs and involves technological risks. One of the basic uncertainties in this relates to choosing a proper integration approach. During the discussions we conducted within the EUDAT consortium (see section 4 for more details), these doubts were confirmed. In addition, like the current solutions provided by EUDAT, integrating object stores into existing community workflows which are still mainly based on the use of filesystems would necessitate significant modification of the workflows.

Therefore, during the second year of the project, we continued the **analysis of and research into object storage technology with a special focus on the** standards and interfaces related to object stores and the possibilities for integrating object stores into the data management systems, repositories and distributed data access solutions already extant in EUDAT. We also reviewed the possible approaches for using object stores as reliable and scalable storage layers in distributed applications and systems and services, including in industry market products and R&D projects. This section summarizes the results of this research. Object Stores, while being increasingly adopted in the main stream (particularly in cloud storage systems), are still a relatively young technology and, as such, are still evolving. New features are constantly being added in response to user demand and increasing adoption of the technology. Many of these new features are directly applicable for use within a distributed, rule-based federated system such as EUDAT and this section will look at some of these new technologies which could have a major impact on what is adopted. However, the features that are detailed in this section are only a small subset of the new features that are available – for a full list one should consult the relevant application pages^{4,5}.

We believe that these new developments and wider technology trends should be taken into account while designing the future architecture of EUDAT services so they will be capable of exploiting the potential of the most modern systems.

2.1. Volume Reduction Technology

One of the greatest recent advances in object storage systems is enabling and increasing support for data volume reduction technologies. While erasure coding was enabled in most popular open source implementations before 2015, we only mentioned erasure coding support in a general sense in the first year M12 deliverable. However we will provide more insight into impact of these mechanisms their usability for CDI at this point.

A common complaint previously levelled at object stores was that, although they allow users to make better use of the available storage by eliminating the need for i-node partition, in order to achieve the same level of data security as that which was typical with RAID systems⁶, it was necessary to create at least two, and preferably three, copies of any data. This ensured that if any of the copies became corrupt, it could be

⁴ <https://releases.openstack.org/>

⁵ <http://docs.ceph.com/docs/master/releases/>

⁶ RAID – Redundant Array of Independent Disk

checked against the remaining copies and replaced automatically if necessary without needing to rebuild the file system. However these additional copies naturally occupied additional space in the storage system and thus created significant additional cost. Such limitations made it not profitable to apply object stores in many use cases from the economic point of view. For instance, using object stores in research programs was impossible as, in such applications, storage infrastructure was funded based on the expected data volume.

In many commercial systems, this was rapidly identified as a significant barrier to the uptake of object stores, even with the concomitant reduction in the effort needed for file system maintenance, and the added ability to store parity information in two additional locations. In the case of the DDN WOS⁷ appliance, it was possible to have distributed parity information for disaster recovery, with the data being held at one site while two copies of the parity information could be held at two alternate sites. While this significantly increased the time needed to rebuild an object, rather than requiring triple the storage capacity, it was now only necessary to provide 1.6 times as much storage (or even less if a single parity copy was required). In this way, the WOS mimicked a distributed RAID-6 system in terms of the levels of data redundancy which that provided.

The two most common open source object stores, Ceph and Openstack Swift now also support erasure encoding of data as an alternative to multiple replicas. In Swift, this was introduced in the Kilo release (2015), while in Ceph it was introduced in Firefly (2014), but was not fully supported until Infernalis (2015). In both cases, erasure coding was included in a highly flexible and configurable manner, which allows a single instance to support both replication and erasure encoding on a pool basis. Thus, depending on the use cases, pools can be set up with replication (for example if the intended use is storage of many small files and frequent access to them or in case if Ceph pool is used as a back-end of the RADOS block devices⁸) or erasure encoding (suitable for storing fewer large objects with infrequent access), or can be configured based on costs.

During 2015 and 2016, erasure coding implementations were stabilized and debugged. Also users started to adopt these solutions in production systems, which resulted in greater availability of experience and knowledge related to erasure coding reliability and performance, as well as the cost efficiency impacts.

On the other end, software, CPU and server producers noticed that there was a need to support algorithms used in erasure coding implementations in programming libraries and CPU instruction sets. Following this trend, Intel included erasure coding functionality in its Storage Acceleration Library⁹ and in its chips. The open source community also developed several libraries such as KODO¹⁰ or jErasure¹¹. Interestingly, there are efforts on porting these libraries to non-Intel x86 compliant platforms such as ARM CPU-based ones. Moreover, companies that build Ceph-based storage appliances, and RedHat (which maintains Ceph), are also investing in optimizing Ceph to implement erasure coding effectively using low-energy platforms. This is now regarded as being profitable as several ARM makers started to include erasure-coding-related instruction sets within their dedicated I/O support units. The low-power-CPU-related developments may make it possible to reduce the purchase and maintenance costs of object stores further, while keeping performance and reliability at the desired level. ARM CPUs are, relatively speaking, cheaper to acquire than Intel CPUs and feature lower energy consumption per CPU with an overall higher number of cores per unit, which facilitates great parallelism with I/O processing. Analysis shows that while, per CPU, ARMs are still slower while implementing erasure coding than Intel x86 CPUs, the overall cost of systems using ARMs might be lower, yet at the same level of I/O efficiency and data security.

Overall, support for erasure coding is a useful feature of object stores that can be used to build cost- and efficiency-optimized configurations. For instance, an object storage-based B2SHARE implementation could apply erasure coding within particular sites, while using full data replication across geographical locations. Such a configuration would make it possible to achieve resilience to small-scale local outages, such as disk

⁷ DDN Web Object Scaler - <http://www.ddn.com/products/object-storage-web-object-scaler-wos/>

⁸ <http://docs.ceph.com/docs/giant/rbd/rbd/>

⁹ <https://01.org/intel%C2%AE-storage-acceleration-library-open-source-version>

¹⁰ <http://docs.steinwurf.com/kodo/index.html#kodo>

¹¹ <https://github.com/tsuraan/Jerasure>

drive or server failures, while keeping the possibility to recover sites in case of a large-scale disaster. While such a configuration could have been achieved previously with traditional RAID5 or RAID6 systems used at data centres and iRODS-based geographical replications, in the case of object stores, local and geographical data redundancy can be implemented in a unified way, based on the integral mechanisms of the object stores.

2.2. Interfaces and Standards

The interfaces and standards related to any distributed software and hardware infrastructure are an important part of the infrastructure. In the first deliverable for this task, we mentioned several basic APIs and standards in the area of object storage interfaces. In this deliverable, we provide more insight on selected aspects of this that we find necessary to take into consideration while planning the future architecture of the EUDAT CDI, including the object stores systems.

2.2.1. Standard Interfaces and their Developments

Amazon S3 protocol¹² remains a de-facto standard interface for the object stores supported by most of the open source and commercial implementations. Its two fundamental features make it very suitable for integrating the object stores into the existing software stack, as well as for exploiting the power of their distributed architecture. First, S3 is based on HTTP, which is well understood in the IT industry with tools, libraries and know-how available in the market. It also proved to work for transferring massive volumes of data across the Internet and serving multiple parallel I/O operations (such as, for example, portals, and content serving platforms). Second, S3 is a RESTful¹³, state-less interface, which helps when it comes to scaling the storage system implementation due to the stateless nature of the interaction with clients.

While particular object storage implementations provide their own native protocols, S3 is widely supported by open source products, including Ceph (through a RADOS gateway) and Swift (through an S3 proxy), as well as in the commercial systems. This includes software-only packages, e.g. EMC ECS¹⁴, NetApp Storage Grid¹⁵, as well as storage appliances such as the DDN Web Object Scaler¹⁶ or EMC Isilon¹⁷.

It is also noticeable that adoption of the OpenStack Swift API has broadened recently. Swift is another object storage interface standard that is semantically close to the S3 API¹⁸. However it has a fully open specification and its implementation in OpenStack Swift is open source, which is one of the reasons for its growing popularity. It is also related to the OpenStack initiative and independent of any particular vendor – that is the opposite of S3 which, while widely adopted, is in fact established and controlled by Amazon.

Support for the Swift API in the systems beyond OpenStack Swift itself is extending. While it has been present in Ceph (through a relevant RADOS gateway), as well as in EMC ECS and DDN WOS, it has recently been added to several products, including the IBM GPFS Spectrum Scale¹⁹.

CDMI²⁰ is another open protocol that could act as a front-end to object stores. CDMI makes it possible to access both classical file system and object-storage interfaces in a unified way. Beside the actual data storage and access, CDMI provides an extensive set of definitions of structure and semantics of meta-data. It also defines vocabularies and semantics for control and management information related to storage systems, containers (aka folders, buckets) and data objects. CDMI is an industry-developed ISO standard²¹ promoted

¹² <http://docs.aws.amazon.com/AmazonS3/latest/API/Welcome.html>

¹³ https://en.wikipedia.org/wiki/Representational_state_transfer

¹⁴ https://www.emc.com/techpubs/ecs/ecs_s3_supported_features-1.htm

¹⁵ <https://www.netapp.com/us/products/data-management-software/object-storage-grid-sds.aspx>

¹⁶ <http://www.ddn.com/products/object-storage-web-object-scaler-wos/>

¹⁷ <https://www.emc.com/collateral/data-sheet/h14758-ds-isilon-cloudpools.pdf>

¹⁸ <http://developer.openstack.org/api-ref/objectstorage-v1.html>

¹⁹ https://www.ibm.com/support/knowledgecenter/en/STXKQY_4.2.0/com.ibm.spectrum.scale.v4r2.ins.doc/bl1ins_objectoverview.htm

²⁰ <https://www.snia.org/cdm>

²¹ <https://www.iso.org/standard/70226.html>

by SNIA, the Storage Networking Industry Association, which is in contrast to S3, being the de-facto standard, developed and owned by Amazon. Therefore one would expect CDMI to be widely adopted in storage products. However, in actual fact, the level of CDMI adoption in the commercial world is very low. Its support in open source products is also very limited: exceptions are the reference implementation of CDMI by SNIA²², and several projects such as CDMI-Proxy²³, CDMI-Serve²⁴ and CDMI for OpenStack Swift²⁵. Most of them, however, are not developed and maintained.

The INDIGO-DataCloud EU project invested in adopting and enhancing the CDMI protocol in order to deal with QoS features of storage systems^{26, 27}. Proposed extensions to the standard are currently being considered via consultation with working groups in the Research Data Alliance and SNIA. The standardisation process is in preparation. However, even in this case, the usage of CDMI is limited to handling metadata and controlling data management, while actual data access and storage (the so-called data path) is in fact implemented using another means, specific to the particular data storage systems, e.g. pNFS for dCache, a local filesystem interface for HPSS, and direct RADOS/S3-based access for Ceph/Swift. The INDIGO-DataCloud project prototyped the extended CDMI-QoS server based on the reference implementation of CDMI provided by SNIA.

The primary issue with these standards, which make wide adoption of object storage technology difficult for science, is that there currently exist significant quantities of legacy codes which are based on POSIX, with little or no effort available for adapting these to new storage technologies. This is an issue across a wide range of research communities, from astronomy (for example, AIPS++, or FITS) to seismology. While many of these codes are open source, some (particularly those developed by commercial sensor providers) are not, and these providers do seem willing to adapt to object storage APIs.

Overall, as there is no widely adopted standard other than S3 and Swift, object storage systems are supposed to provide S3 as a mandatory integration interface, with optional support for Swift. While designing the integrated and federated storage systems and applications, support for S3 back-ends should be considered mandatory. In addition, OpenStack Swift support should be added wherever possible and affordable for the sake of its openness and independence of any particular vendor.

2.3. Object Storage Integration Options

The integration possibilities of the object stores are an important aspect of the research on future CDI architecture. In T9.1.1, we continued to analyse the latest developments in this area.

2.3.1. Filesystem-like Interfaces to Object Stores

In part to overcome the inability to adopt legacy software systems to object store APIs, a noticeable trend in the storage software and services market is to provide file system-like access to object storage systems and services. While S3, and other APIs for object stores, are increasingly adopted in new software stacks, existing applications and systems handle data access mostly through POSIX or POSIX-like interfaces. Therefore there is a need to enable these legacy applications and systems to access and store data in objects stores, in order to ensure their compatibility with modern technologies.

This also applies to the elements of the EUDAT CDI software stack, including B2SHARE, B2SAFE and B2DROP. While an S3 plugin is under development, it has not yet been officially published for the version of Invenio²⁸ that is the basis of B2SHARE. Therefore, file system to S3 translation mechanisms must be used for storing

²² <https://github.com/SNIA/CDMI>

²³ <https://github.com/livenson/vcdm>

²⁴ <https://github.com/koenbollen/cdm-serve>

²⁵ <https://github.com/osaddon/cdm>

²⁶ Millar, Fuhrmann, Brzezniak, Hardt, Ertl. Storage Quality-of-Service in Cloud-based Scientific Environments: A Standardization Approach, 10-14 Oct 2016, 22nd International Conference on Computing in High Energy and Nuclear Physics (CHEP2016).

²⁷ <https://www.dcache.org/manuals/workshop-2017-05-29-Umea/000-Final/anupam-cdm-qos-v02.pdf>

²⁸ <http://invenio-software.org/>

and accessing the digital objects deposited in the repository to and from object stores. Similarly, while B2DROP could use object stores at the back-end through the nextCloud S3 or Swift plugin²⁹, sharing the B2DROP-managed data with other EUDAT services that still mainly use the POSIX interface would be problematic. A similar problem appears for iRODS, which is still the central technology for B2SAFE. On one hand, it could use object stores directly through its S3 resource plugins^{30,31}. On the other hand, accessing object stores through a file system-like interface would have the benefit of sharing the back-end with POSIX-speaking components that are still dominant in the EUDAT CDI.

During the second year of task 9.1.1 we have carefully analysed available solutions from both architecture and implementation perspectives. The results of this analysis, and observations on the currently available and future options for integrating object stores into the data management software stacks, are summarised below.

Compatibility Issues

The first problem is that most of the legacy applications and systems expect storage to use the POSIX protocol and to implement file system-like semantics. This includes the ability to represent hierarchical data structures and relationships, such as directories, sub-directories and files. Also providing transactional consistency guarantees is typically expected, as they are natural in filesystems. Directory-level and file-level locking would also be required in several use cases. However, these functionalities and behaviours are not typically present in object stores.

Therefore most of the file-system to object storage translation solutions are limited in their ability to mimic full file system semantics. They are mostly suitable for sequential access to whole files and objects, rather than for performing random I/O, modifications of fragments of objects, or handling massive file system-level operations, such as object renaming or moving objects between directories.

Second, the end users are typically not familiar with the specifics of object storage data organisation and they therefore expect file system-like representation of the structure of the data sets. S3 supports the concept of folders, which can give a hierarchical view of the objects stored within a bucket, but does not follow POSIX semantics. Several alternative solutions make it possible to ‘mimic’ the hierarchical data organisation within the in-reality flat containers while presenting a POSIX, or POSIX-like, syntax. For example, S3FS³², a client-side virtual file system that allows access to S3 buckets, emulates the pseudo-directories by storing the objects with the relevant name³³; a similar approach is adopted in the iRODS plugin for object stores; however, using such a simplistic file system to object storage translation mechanism without making users aware of its limits may lead to errors and data inconsistency.

The third problem relates to the mapping of files into objects. Some solutions allow one-to-one mapping of files to objects. In this class of systems, each file in the file system-like representation is stored as an object in the object store. With the addition of a namespace mapping mechanism, this makes it possible to access data objects using file system and object storage interfaces without applying complicated translation logic. However, in several solutions, files are in fact stripped and spread (and optionally replicated or erasure coded) across the object storage cluster for performance and reliability reasons. In such cases, accessing data relevant to the files through the object storage interface is more complicated as the data organisation in the object store is different. In our analysis, we focus on the systems where one-to-one mapping is possible, as it makes it possible to access data objects through file system-like interfaces for compatibility purposes, while enabling usage of native object storage protocols for high-performance data storage and access.

²⁹ http://www.tengen.com.tr/en/documents/Datasheet_enterprise_and_subscription_ENG.pdf

³⁰ https://docs.irods.org/4.2.0/plugins/composable_resources/#amazon-s3-archive

³¹ https://github.com/irods/irods_resource_plugin_s3

³² FUSE-based file system backed by Amazon S3. <https://github.com/s3fs-fuse/s3fs-fuse>

³³ in S3FS file “d.txt” within a virtual path “/a/b/c/” is actually stored as object named “/a/b/c/d.txt”, while the virtual directory “/a/b/c/” is not represented in the object storage namespace; while S3fs can emulate existence of the path “/a/b/c/” to some extent, e.g. by enabling “listing” its contents, full support for filesystem operations is not possible, e.g. all objects stored under /a/b/c cannot be atomically moved to another path; also the virtual path rename operation would in fact require would in fact require renaming all objects “contained” in this path

Representatives of this group include, amongst others, S3FS, ProxyFS and the NFS gateway for Ceph (see details in the following section).

Architecture and Implementation Issues

Another set of problems results from the architecture of the file system to object storage translation solutions and their implementation specifics. Architecture-wise, two main groups of solutions are available: client-side mechanisms and storage system-level features.

Client-side solutions are typically not scalable enough to serve large data volumes and complicated data structures. For instance, the most popular solution, S3FS, has several scalability limitations that are representative of this class of object storage to file system translation solutions.

First, due to the specifics of its implementation, S3FS introduces latency into the data path as the user application data traverse the user and kernel space boundaries several times while processing each I/O request. Moreover the translation mechanism cannot be parallelized, at least from the perspective of a given client interacting with a given bucket or container in the object store. While for the human user high latency and I/O path centralisation can be acceptable, such an approach may be problematic for large-scale data handling, such as repositories, portals or other applications that process and serve the digital content from object stores to the users over the Internet.

Second, the translation mechanisms constitute an intermediate step in the application-storage interaction. Even if they were implemented using a kernel driver, the additional latency introduced by its logic might be unacceptable in many use cases.

Third, the filesystem to object storage translator specifics can limit the actual usability of the mechanism. Many of the existing tools in fact enable access to only one container or bucket on the object storage system side. This includes S3FS and Cyberduck³⁴. Although this approach is feasible for simple use cases, such as accessing or storing data objects in a sequential manner, it may cause severe scalability or reliability issues on the larger scale and in more complicated use cases. For instance, putting many files into a single container in Swift may impact the client I/O performance or the complexity of the internal processes resulting in increased latency when it comes to serving user I/O requests or excess consumption of resources on the system (as containers are periodically replicated using rsync, with a growing number of objects, the rsync process consumes more resources and thus impacts the client I/O performance).

Overall, client-side file system to object storage translation mechanisms should be considered as convenience features rather than target solutions. While part of the problem results from the implementation specifics and can be solved with reasonable effort, such mechanisms share the issues resulting from their architectural limits. Other approaches are required in order to fully exploit the power of scalable object storage systems.

Server-side solutions, which implement the translation mechanisms on the storage system side, potentially offer a more scalable and elegant solution. The software market already provides products in this category.

ProxyFS³⁵, a native (that is, object storage side) implementation of a file system to object translation mechanism for OpenStack Swift is one such example. It is implemented as one of the OpenStack Swift proxies and runs on the same level of system hierarchy as Swift proxy and S3 proxy. ProxyFS can use internal mechanics and APIs of the Swift cluster in order to implement its functionality, which is the opposite of the gateway-based approaches that are limited to using object storage public APIs only. ProxyFS also has full awareness of the state of the cluster and processes happening in the cluster and has direct access to the Swift Storage Nodes. This makes it scalable and suitable for large-scale deployments. File system-like access to objects is possible from the ProxyFS node, which makes it possible to serve the object storage content through NFS or other POSIX-like protocols. While ProxyFS is not an open source solution, it is planned to be officially available in the version of Swift offered by SwiftStack; the fact that the company invested in this

³⁴ Cyberduck. Libre FTP, SFTP, WebDAV, S3, Azure & OpenStack Swift browser for Mac and Windows. <https://cyberduck.io/>

³⁵ <https://www.openstack.org/videos/austin-2016/file-this-swift-api-then-s3-api-and-now-posix-access-to-openstack-swift>

project confirms that the market recognizes the need to provide file system-like access to object stores at the proper level of scalability and reliability.

Another example from the open source area is the NFS gateway for Ceph^{36, 37}. It provides NFS services on top of the Ceph object storage cluster accessed through Ceph's native RADOS protocol. It is based on a user space implementation of the pNFS server (NFS-Ganesha³⁸) and the RADOS client library (librados³⁹). Architecturally, the NFS gateway sits at a similar level of hierarchy as the S3 and Swift gateways for RADOS that make HTTP access possible. Functionality-wise the NFS gateway enables basic POSIX-like access to data stored in the RADOS cluster, such as sequential writing to the object store, as well as sequential and random/partial reads from the objects. The NFS gateway lacks support for several NFS and POSIX semantics including renaming directories, supporting NFS ACLs and symbolic and hard links, but offers a good level of functionality compatible with many legacy applications. Similar to other file system to object store translation mechanisms, the NFS gateway should be used for data ingest/retrieve to and from object store rather than for random, concurrent storage and access, and it is not intended to replace the fully-fledged NFS service based on cluster file systems. Despite its functional limitations, the Ceph NFS gateway is a useful tool facilitating backward compatibility of object stores with the current software stacks for data management that still assume file system-like data organisation. It is important to note that the NFS gateway is an integral part of the mainstream distribution of Ceph (starting with the Jewel version) which shows that file system to object storage translation is considered an important component of the storage software ecosystem.

In parallel with the large open source initiatives, academic institutions and R&D projects are working on integrating object stores into data management environments. STFC, a R&D centre in the UK which is also an EUDAT partner, developed the GridFTP⁴⁰ to Ceph RADOS gateway (gridFTPCephPlugin⁴¹). It is implemented as a plug-in to the GridFTP server which has a modular architecture that makes it possible to extend it with third-party plug-ins. The GridFTP gateway enables end-users and data management agents to use GridFTP, an extension of the FTP protocol developed for the grid computing community, in order to perform high performance, multi-threaded and secure transfers of large files to and from the Ceph clusters. While the plug-in is in an early version, and several issues arise during its usage (such as sensitivity to the version of Ceph cluster or client library), it is a promising solution for efficiently integrating object storage systems into existing scientific software stacks. While it does not address all the possible issues and use-cases while integrating POSIX and object stores within an infrastructure, it contributes to the wider trend of providing compatibility between legacy applications and systems, and modern, object storage-based solutions.

Work is also currently being undertaken to allow the common HDF-5/netCDF formats to make better use of object stores, including porting the APIs to support Swift and S3. While at least one example of this is already available (h5⁴²), additional work is being performed within the core HDF5 development team (personal communication). If this work is successful, applications already making use of this common format will be able to make direct use of object stores or normal file systems without any change to the existing code base. This is yet another indication that the advantages of object stores are now sufficiently well established that some common legacy applications are willing to adapt to them.

Industry also recognizes the necessity and benefits of exploiting the power of object storage systems on one hand, while providing file system-like access to the storage service on the other. For instance, EMC GeoNAS provides network file services such as NFS and CIFS based on ECS clusters at the backend. ECS is an object storage implementation developed by EMC. GeoNAS can also use other object stores, including the Amazon, Google and Microsoft public clouds. While GeoNAS is a closed-source, vendor-provided solution (and thus

³⁶ <http://docs.ceph.com/docs/master/radosgw/nfs/>

³⁷ <http://ceph.com/planet/ceph-rados-gateway-and-nfs/>

³⁸ <https://github.com/nfs-ganesha/nfs-ganesha/wiki>

³⁹ <http://docs.ceph.com/docs/master/rados/api/librados/>

⁴⁰ <http://toolkit.globus.org/toolkit/docs/latest-stable/gridftp/>

⁴¹ <https://github.com/ijiorama/gridFTPCephPlugin>

⁴² <https://cran.r-project.org/web/packages/h5/vignettes/h5-Intro.html>

the wider public cannot really benefit from it), the fact that EMC and other companies are working on file system-like access to object-storage systems confirms that this approach makes sense.

2.3.2. Native or Standard Interfaces-based Integration

The full potential of object stores can be exploited while using their native interfaces such as the RADOS client library for Ceph, Swift API for OpenStack Swift or standard web-based REST interfaces such as S3. In contrast to using file system to object store translation, direct usage of object storage APIs may require the redesign and adaptation of the I/O logic in existing applications, as well as adopting new paradigms while designing new systems. However the effort needed to accomplish full object storage integration brings many benefits.

First of all, by using object stores APIs directly from the application, there is no need for translating the data and meta-data representations between the file system-like organization and object storage, which is costly in terms of performance, scalability and complexity. Similarly, simulating POSIX semantics is not always necessary. Also, caching, as well as creating temporary representations of data objects in memory or on disk or in the file system cache of the gateway systems, is not necessary.

Second, the applications and services can use object storage back-ends at their full speeds. Native APIs are typically optimized for bandwidth and latency. They also support parallelism of communication towards a given endpoint and the storage cluster as a whole.

In particular, Swift enables clients that use Swift or S3 API to communicate with an arbitrary Proxy Node that routes the traffic to selected Storage Nodes. While writing the minimum number of object replicas to Storage Nodes is performed synchronously, writing of the data on the target redundancy level is performed asynchronously. This enables clients to continue their operations without waiting for confirmation that all replicas or redundant information have been saved successfully. Moreover, applying HTTP-level load balancers in front of the Proxy Nodes can improve parallelism of the client access to Swift cluster.

In the case of Ceph, parallelism is achieved thanks to the following facts. Particular clients can speak to multiple storage components, i.e. OSD daemons (typically assigned to disk drives) based on the information on the RADOS cluster state acquired from Ceph Monitors and the content of the CRUSH map. OSDs⁴³ are assigned to clients independently for each object to be written or accessed.

Moreover, client I/O request can be served independently by multiple OSDs. Thanks to such approach, high efficiency of writing can be achieved despite the fact that while serving write I/O operations OSDs may require a certain number of data copies (replicas) or a certain volume of redundancy information (erasure coding) to be written synchronously, before they confirm successful write operation to the client.

Finally, Ceph clients, including RADOS gateway that provides S3 and Swift API as well as CephFS and RADOS Block Device (RBD) clients, can stripe (i.e. distribute)⁴⁴ their data over multiple objects in the Ceph storage cluster. This allows a yet higher level of I/O parallelism.

It is important to note that both open source implementations of object stores enable parallel access to the storage cluster for S3. While one may expect some overhead with the S3 gateway in Ceph, compared to librados-level performance, S3 Proxy for Swift works on the same level of abstraction as the native Swift Proxy.

Overall, while exploiting the low-level native APIs of object stores may have advantages in certain cases, binding the user applications or data management systems to particular APIs may become problematic in the long term and make it problematic and costly to guarantee the compatibility of the applications with potentially changing object storage technologies. On the other hand, the implementations of standard APIs for most object stores are well optimized. Therefore using S3 or Swift APIs for object storage systems is

⁴³ OSD is a physical or logical storage unit in the Ceph cluster. Typically one OSD is defined per one physical disk drive.

⁴⁴ Refer to „Data Striping” section of the <http://docs.ceph.com/docs/master/architecture/> document

considered a safe and reasonable approach ensuring a good balance between efficiency, reliability and the complexity of developing and maintaining application and services software stacks.

Client libraries for S3 are the most widely available, including language bindings for Python, Java, C/C++, JavaScript and many others. Swift API clients are now available for most of the programming languages including Python⁴⁵, Java, C/C++, Ruby and Go⁴⁶. Ceph librados implementations are provided for C/C++, Python, Java and PHP⁴⁷.

2.4. Market Trends Related to Object Storage Integration

In this section we overview the trends related to the adoption of object storage in market products related to long-term storage, data management, and data repositories, as well as data synchronization and sharing. This information provides a background for our considerations related to the adoption of object stores in EUDAT.

2.4.1. Long-term and Large Scale Storage

It has already been pointed out several times that object stores are increasingly important components of today's IT infrastructures. Their architectural potential and maturity of implementation make them interesting candidates for efficient, large-scale, reliable and long-term data storage.

As part of this task we analysed the latest developments in the software and services market, as well as in the R&D area, related to the adoption of object stores and approaches for integrating them into data management applications and systems. A summary is provided in the following points which cover exemplary applications that are addressing aspects of data management that are relevant for EUDAT services.

2.4.2. Long-term Storage and Large-scale Storage Management

Providing long-term, reliable storage of registered data is one of the fundamental functionalities of the EUDAT CDI. This is provided via the B2SAFE service, which is one of the flagship services of the EUDAT project and infrastructure.

In this section we review data management products and R&D initiatives that target this application area and exploit the object storage paradigm or use object storage systems as the data persistency and access layer.

The Amazon Simple Storage Service (Amazon S3) is the pioneering product in this context. It is a public cloud service that provides data storage and access functionality through its web-based REST API⁴⁸ which became a de-facto standard in the market. Internally it is solely based on the object storage concept and architecture. It also demonstrates functionality and features that are considered as a reference for object storage systems.

In addition to simple storage services, including depositing and accessing data objects, Amazon provides data management-related functionality. The service includes support controlling the quality of service of storage by assigning storage classes to the storage buckets⁴⁹. These classes are characterized by assumed access patterns including Standard, Standard - Infrequent Access, Archive - Glacier. Users may also decide where their data is stored (that is, they can choose among the available Amazon regions⁵⁰) as well as defining cross-region replication of buckets. There is also support for defining simple policies that steer replication, as well as moving the data between storage classes based on data life-cycle related conditions (such as the creation of new object versions, object deletion, expiration, reaching a particular age or reaching the threshold related to the frequency of the access)^{51,52}. Monitoring of the replication status is also possible⁵³.

⁴⁵ <https://github.com/openstack/python-swiftclient>

⁴⁶ <https://wiki.openstack.org/wiki/SDKs>

⁴⁷ <http://docs.ceph.com/docs/hammer/rados/api/librados-intro/>

⁴⁸ <http://docs.aws.amazon.com/AmazonS3/latest/API/>

⁴⁹ <https://aws.amazon.com/s3/storage-classes/>

⁵⁰ <https://aws.amazon.com/about-aws/global-infrastructure/regional-product-services/>

⁵¹ <http://docs.aws.amazon.com/AmazonS3/latest/dev/crr-how-setup.html>

⁵² <http://docs.aws.amazon.com/AmazonS3/latest/API/RESTBucketPUTreplication.html>

⁵³ <http://docs.aws.amazon.com/AmazonS3/latest/dev/crr-status.html>

Functionality similar to that of Amazon S3 is implemented by several cloud data management services such as Google Cloud Storage, IBM SoftLayer, and Microsoft Azure storage. While protocol details and implementation specifics are different, a common feature of these systems is the adoption of the object storage paradigm and HTTP REST APIs.

Amazon S3 and similar services can be perceived as the long-term storage and large-scale data management solutions. Their basic functionality is also exploited and integrated into many data management and long-term, large-scale products on the market and also by academic R&D and infrastructure projects.

There are various approaches used to integrate object storage into high-level services for long-term storage and data management. Among the fundamental features of these systems is how multiple storage systems are combined and presented to the users, as well as the impact of the integration mechanism on the data path from the client to the storage service. In the following subsection we review several data management products and solutions in order to present fundamental features of the possible approaches to object store integration, from the architectural point of view, and to provide a short analysis of them in the context of EUDAT needs.

In-band vs Out-of-band Integration

DuraCloud is a solution for unifying access to different cloud storage services widely used in the US for building content preservation systems. It is a Web-based service that makes it possible to store data in Amazon S3 and several other storage systems, including Amazon Glacier, RackSpace Cloud Files and the San Diego Supercomputing Center Cloud Storage⁵⁴. Clients interact with DuraCloud servers (DuraStore), using a custom Web-based API (DuraCloud REST API⁵⁵). Data storage and access, as well as data management requests, are routed through the DuraStore servers to particular storage providers and translated through a set of storage adapters⁵⁶, including an S3 storage adapter implemented using the standard Amazon S3 Java client library. In addition to I/O routing and requests translation, DuraStore performs several data management functionalities, such as storage mediation, data integrity checking, and restoring proper versions of corrupted or missing files.

From the architectural point of view, DuraStore servers act as the abstraction and translation layer for the underlying object stores. Importantly, DuraCloud works on the data path from the client to the storage system. Therefore it introduces potential bottlenecks due to I/O traffic centralization and latency due to protocol translation for the benefit of unifying access to various storage systems and providing data management functionality. Notably, from the perspective of the data path, DuraCloud is similar to iRODS which is a central component that is able to translate custom protocol queries to the underlying storage system's language.

iRODS integrates object stores through its S3 resource plugin^{57,58}. This feature has been available since 2014. It uses the C/C++ client library for S3⁵⁹, and is able to work with the Amazon S3 service and with private S3 instances. Based on the iRODS federation mechanisms with help from iRODS rules, it can also perform replication of the data between multiple S3 clusters located in the same data centre or separated geographically. Within the first phase of the EUDAT project we performed an analysis of the mechanism for integrating S3 with iRODS, which confirmed the usability of the plugin for simple use cases. However it also revealed several issues that arise while simulating POSIX-like logic on top of object stores that provide limited semantics, including namespace mapping, and handling renames. This experience led to the conclusion that the object names should perhaps be logically decoupled from the iCAT level file names that are presented to users.

⁵⁴ <https://wiki.duraspace.org/display/DURACLOUDDOC/DuraCloud+Storage>

⁵⁵ <https://wiki.duraspace.org/display/DURACLOUDDOC/DuraCloud+REST+API>

⁵⁶ <https://wiki.duraspace.org/display/DURACLOUD/DuraCloud+Architecture>

⁵⁷ http://slides.com/jasoncoposky/s3_architecture#/

⁵⁸ https://github.com/irods/irods_resource_plugin_s3

⁵⁹ <https://github.com/irods/libS3>

The dCache⁶⁰ team recently came up with several developments related to object storage adoption. Although Ceph is integrated through the RBD client for now, the work that was performed makes it possible to integrate object stores directly using their native protocols and enables interfacing to DDN WOS, Swift, S3 and CDMI, according to dCache developers⁶¹. dCache creates a mappings of the user-level namespace to the storage-side file or object names, stores these mappings in the Chimera database, and applies them while serving I/O requests towards data objects. This provides flexibility while integrating various storage back-ends. In the case of object stores, this mechanism helps to address the incompatibility of the file system-like logic and POSIX-like namespace with the object storage. For instance, operations such as renames can be performed by updating the metadata database entries, and no complicated logic is required to handle multiple object name changes or moving objects within the hierarchical structure.

While iRODS and dCache differ in architecture and implementation, their common feature is that, from the architectural point of view, they may be perceived as gateways among the non-S3 clients and S3-storage systems. Therefore, architecture-wise they are central components on the I/O paths and may constitute performance bottlenecks at a large scale.

On the other hand, there is DynaFed⁶², an HTTP-based solution for integrating multiple storage technologies and end-points “into a transparent, high performance storage federation that exposes a unique name space”⁶³.

Fundamental feature of DynaFed is that it implements the federation functionality out of band, as it uses I/O request redirection extensively. In particular, for metadata queries (such as listing the directory content) it works as an in-band proxy employing the local metadata cache. However, for storing or accessing the actual files or objects, it exploits HTTP protocol redirection – that is, it points the clients to the storage end-points that hold the copies of the requested files or objects or is able to accept new data. The selection of the actual site for data access or storage is based on a pluggable mechanism that makes it possible to select replicas among other based on geo-IP mechanism. Clients contact the HTTP/WebDAV or S3 endpoints directly in order to perform I/O and the federation components are not involved at this stage.

The approach adopted in DynaFed makes it possible to integrate object stores with only minimal impact on the I/O performance and with high reliability. Performance-wise the federation is lightweight as it is based only on metadata discovery (limited to areas pointed to by the client’s queries) and its in-memory caching as well as I/O redirects. Reliability of the mechanism is achieved by running multiple instances of DynaFed along with HTTP load-balancers. Thanks to their stateless nature, the federation components can be failed-over without additional delays as they do not contain any information that has to be protected and passed among instances, apart from the cached meta-data that can be recovered from the storage end-points.

Importantly, DynaFed has the potential to implement object storage federation in a fully transparent way. While the official version of DynaFed supports only HTTP/WebDAV at its front-end, efforts are being made (notably within EUDAT task 9.1.2) to provide it with an S3 interface towards the federation clients. This will enable fully S3-speaking clients to interact with multiple object stores that will appear as a logical uniform S3 endpoint.

While DynaFed does not implement data management tasks itself, it presents an interesting approach to integrating object stores and other types of storage systems with high-level data services. DynaFed can be combined with external components that implement data management functionality, such as replication, consistency checking, and PID registration. For instance, in CERN’s WLCG project, DynaFed is complemented by FTS⁶⁴, which is used for data transfers related to replication, while DynaFed itself provides federated and transparent access to data sets⁶⁵. The built-in capability of the object storage systems could be used for

⁶⁰ <https://www.dcache.org/manuals/2013/presentations/dCache-agile-technology-v5.pdf>

⁶¹ <https://www.dcache.org/manuals/workshop-2017-05-29-Umea/000-Final/tigran-dcache+ceph.pdf>

⁶² <http://lcgdm.web.cern.ch/dynafed-dynamic-federation-project>

⁶³ http://svnweb.cern.ch/world/wsvn/lcgdm/ugr/trunk/doc/whitepaper/Doc_DynaFeds.pdf

⁶⁴ <http://information-technology.web.cern.ch/services/file-transfer>

⁶⁵ <https://indico.egi.eu/indico/event/2615/material/slides/1>

guaranteeing data replication and performing other data management-related actions, such as PID registration. It could be defined on the storage cluster or object storage bucket basis. Moreover, due to DynaFed's dynamic nature, it can benefit from the data management processes. For instance, additional data replicas can be used in order to serve the I/O requests from the clients using the extra data as they appear in particular storage endpoints.

Overall, the above analysis confirms that the adoption of object storage systems in the data management and long-term storage systems is constantly growing. It also shows that new decentralized approaches to I/O handling, routing the client-storage traffic and controlling the data path have to be worked out in order to avoid architecture centralization and to make it possible to exploit the full potential of object stores, especially in a geographically distributed setup.

This need has been observed by organizations including Amazon and CERN that have long-running experience in managing massive-scale data archives and providing efficient and transparent access to them.

We believe that EUDAT should take similar approach while planning the future evolution of the CDI. Its architecture should be shaped following the principles adopted in global-scale systems. In particular, the centralization of I/O paths should be avoided by applying out-of-band federation mechanisms for data access. Also, decomposition of current monolithic architecture of EUDAT services into fine-grained components should be done in order to increase the flexibility of implementing data management services and features needed for long-term data storage and preservation as lightweight, loosely coupled modules.

2.4.3. Data Repository Software

Another fundamental functionality of the EUDAT CDI is the data repository solution, namely the B2SHARE service. This section overviews the status of the adoption of object storage systems in the data repository software and provides an analysis of these systems in the context of EUDAT needs.

The B2SHARE service is currently based on the Invenio⁶⁶ data repository software developed and maintained by CERN. Aside from EUDAT, Invenio is also widely used by several large communities (including CERN) for publishing the data from their experiments, by the INSPIRE project for their digital repository and by several digital libraries.

We had already analysed the possibility of integrating an Invenio-based B2SHARE with object stores in the first year of T9.1.1, however, at that point, the status of the support for S3 back-ends was not clear. By the end of 2016, a new version 3.0 of Invenio had been released by CERN and integrated (by the EUDAT B2SHARE team) into version 2.0 of B2SHARE. While the new version of Invenio does not support S3-based storage yet, interfacing to object stores could be enabled with very little modification (according to the B2SHARE and Invenio developers). Moreover, official support for S3 back-ends is on the Invenio roadmap⁶⁷.

If we look beyond EUDAT, Omeka⁶⁸ is a popular repository software. It has been adopted in several European projects and initiatives, including LoCloud⁶⁹ which provides cultural heritage institutions with ability to run dedicated data repositories according to the public cloud model. While Omeka is targeting relatively smaller communities and non-IT experts, from the point of view of our analysis, it shares architecture principles with Invenio. It supports several Web back-ends through plug-ins, including Internet Archive and SoundCloud⁷⁰, among which InternetArchive provides an S3-compatible API⁷¹. Therefore, Omeka in fact uses S3-speaking storage for actual data object storage and access, while providing added value, domain-specific services on top.

⁶⁶ <http://invenio-software.org/>

⁶⁷ <https://media.readthedocs.org/pdf/invenio/latest/invenio.pdf>

⁶⁸ <https://omeka.org/about>

⁶⁹ <http://www.locloud.eu/>

⁷⁰ <https://github.com/Daniel-KM/BeamMeUpToInternetArchive>

⁷¹ <https://github.com/vmbrasseur/IAS3API/blob/master/compatibility.md>

Dspace⁷² is a popular and comprehensive data repository solution developed and backed by DuraSpace. It is typically adopted by relatively big organisations for building large-scale, comprehensive data repository systems. Dspace can use Amazon S3 storage at its back-end through a pluggable storage mechanism^{73,74}. With proper configuration⁷⁵ it can also use the S3-speaking storage clusters such as Ceph or Swift.

Fedora Commons⁷⁶, also backed by DuraSpace, is another popular solution that provides basic functional components for building data repositories. It has a large community of developers and users – mostly universities in the US, UK and across Europe. It has been adopted by several projects, including Islandora⁷⁷ and Samvera⁷⁸, as a basis for building their own digital repository software. Fedora Commons is known for its scalability, extendible and pluggable architecture, support for domain standards and APIs, as well as comprehensive content exploration features. Among these standard, Fedora supports various storage back-ends, including filesystems and S3-speaking object storage (including Amazon S3⁷⁹ and other systems that provide S3 API).

Overall, in the repository software packages (Invenio, Dspace) and repository building components (Fedora) that have been discussed here, it is a common trend to adopt object storage systems as the content storage and serving back-end. Despite the architectural and technological differences between these products, as well as the different types of audience that they target, the usage of object stores brings similar benefits, including the possibility to delegate the functionality of ensuring data persistency to an external module or service and ensuring the capacity and I/O performance scalability of the back-end storage system beyond the possibilities of local file systems. Moreover, the fact that object stores offer HTTP-based REST APIs makes them a natural and easy to integrate component of content management and serving platforms.

Importantly, the known limits of object stores related to their semantics and the namespace-related issues that result from these limits are not showstoppers when it comes to using object stores in data repositories. Most of the repository software virtualizes the user-level namespace, so user-accessible digital object names or identifiers are mapped to the storage level names anyway for long-term sustainability reasons.

Overall, EUDAT can learn a valuable lesson from the developments of data repository solution presented here. From the long-term perspective, it would be profitable to enable B2SHARE service to use scalable, large-capacity and high-performance object storage back-ends. This can be achieved by investing effort into providing support for the S3 protocol in Invenio or by integrating B2SHARE with future scalable modules of the EUDAT CDI, such as projected HTTP-based federation that will offer efficient access to object stores.

2.4.4. Sync and Share Service

As a complement to the long-term storage and data repository services, EUDAT provides the B2DROP service that supports researchers at the early stage of handling datasets, including exchanging and sharing them with other EUDAT users, as well as automatically transferring data to EUDAT online storage services and to B2SHARE.

B2DROP is based on NextCloud⁸⁰ (and has been since the beginning of 2017; prior that it was based on ownCloud⁸¹). NextCloud is a popular sync and share system, integrated with EUDAT's B2SHARE. The B2SHARE integration is a custom, EUDAT-made extension of NextCloud that makes it possible to publish the files deposited in B2DROP using the B2SHARE REST API.

⁷² <http://www.dspace.org/>

⁷³ <https://wiki.duraspace.org/display/DSPACE/DSPACE+2.0+Pluggable+Storage>

⁷⁴ <https://wiki.duraspace.org/display/DSDOC6x/Storage+Layer>

⁷⁵ <http://dspace.2283337.n4.nabble.com/Using-private-s3-object-store-instead-of-aws-td4682247.html>

⁷⁶ <http://fedorarepository.org/features>

⁷⁷ <http://islandora.ca/>

⁷⁸ <https://samvera.org/>

⁷⁹ <https://wiki.duraspace.org/display/DEV/Fedora+on+AWS>

⁸⁰ <https://nextcloud.com/>

⁸¹ <https://owncloud.org/>

The following features of NextCloud and ownCloud are interesting from the point of view of our analysis. Both NextCloud and ownCloud Enterprise make it possible to integrate Amazon S3 storage and other object stores speaking S3 and Swift for storing the contents of user files. This means it is possible to use a single bucket in the object store as an alternative to file system-based storage. Interestingly, they can also distribute the data across multiple buckets in order to facilitate higher scalability^{82, 83}.

The well-established support for S3-speaking storage enabled in NextCloud and ownCloud makes B2DROP ready to integrate with object stores. However, while this is technically possible, scaling the back-end capacity and I/O capabilities has not yet been given high priority for the B2DROP developers as their recent focus has been on integrating B2DROP with the remaining EUDAT services, including B2SHARE and B2ACCESS.

In the wider market perspective, beyond the EUDAT context, many user communities in academia demanded enabling usage of the object stores at the ownCloud and NextCloud back-ends. This applies to both individual institutions, such as universities, and collaborations, such as NRENs and the Geant project.

This fact confirms that the general audience considers object stores to be a scalable, reliable and cost-effective alternative to the file system-like systems, suitable for implementing large-scale sync and share systems.

Another solution representative for the sync and share market, which is not however used by EUDAT, is Seafile⁸⁴. It is a specialized system designed for reliable and efficient data synchronization and sharing. In contrast to ownCloud and NextCloud, Seafile employs a data model that is conceptually similar to Git, which ensures great data synchronisation efficiency. Seafile can exploit object storage, including Amazon S3 and S3-compatible systems⁸⁵, as well as Ceph⁸⁶ and Swift⁸⁷. Seafile performance analysis at PSNC⁸⁸ confirmed that it has the potential to exploit the high level of I/O parallelism that is offered by object storage systems. Taking into account the cost efficiency of object stores, they are great option for implementing large-scale sync and share services using private cloud-based platforms such as Seafile.

Last, but not least, Dropbox, the globally most popular and de-facto standard sync and share service has used the Amazon S3 storage for years, since its launch in 2008. However, the company recently made an architecture and technology shift⁸⁹ towards their own custom-built infrastructure and block-based Magic Pocket^{90,91} storage. This fact demonstrates the applicability of S3 stores to sync and share applications, at least at the scale of EUDAT. Notably, in 2016 Dropbox still used European regions of Amazon S3 for hosting the data of its customers from German speaking countries, including Germany, Austria and Switzerland⁹². Taking into account the fact that the number of Dropbox users in these areas was in the range of hundred of thousands of people, such a scale is still much larger than that covered by the EUDAT CDI.

⁸² https://docs.nextcloud.com/server/11/admin_manual/configuration_files/primary_storage.html

⁸³ https://doc.owncloud.org/server/latest/admin_manual/enterprise/external_storage/s3_swift_as_primary_object_store_configuration.html

⁸⁴ <https://www.seafile.com/en/home/>

⁸⁵ https://manual.seafile.com/deploy_pro/setup_with_amazon_s3.html

⁸⁶ https://manual.seafile.com/deploy_pro/setup_with_ceph.html

⁸⁷ https://manual.seafile.com/deploy_pro/setup_with_swift.html

⁸⁸ Brzeźniak M., Wadówka K.; Woszek P.; Meyer N. Storage back-ends for Scalable Sync & Share based on Seafile. Proceedings of CS3 2017 at SurfSARA. <https://zenodo.org/record/439564#.WZ7jyndJZE4>

⁸⁹ <https://www.wired.com/2016/03/epic-story-dropboxs-exodus-amazon-cloud-empire/>

⁹⁰ <https://blogs.dropbox.com/tech/2016/03/magic-pocket-infrastructure/>

⁹¹ <https://blogs.dropbox.com/tech/2016/05/inside-the-magic-pocket/>

⁹² <https://blogs.dropbox.com/business/2016/02/dropbox-is-growing-in-europe/>

3. DISCUSSIONS AND FEEDBACK ON THE CONCEPT

The aim of WP9 as expressed in the project's DoA was to “strike a balance between ambitious, far-reaching goals and direct applicability and immediate relevance of the solutions.” Therefore, we considered the consultation process to be an important part of the work of our task that aims at envisioning and designing the components of the future, scalable EUDAT CDI.

Therefore, based on the study of the applicability of object storages within CDI and the integration options documented in the M12 deliverable in the second year of the project, we conducted a series of discussions within the EUDAT project and beyond, related to the concepts and proposals worked out by us.

In this section we overview these discussions and the feedback that was acquired as a result of them. This information gives important input into the architecture recommendations included in Section 5 and the final conclusion of our work, which is covered in Section 6.

3.1. Discussions within EUDAT

An obvious environment in which to discuss our results and proposals is the EUDAT consortium. The main part of the discussions happened within our JRA, however we also discussed the adoption of object stores and the potential evolution of the EUDAT CDI architecture with various stakeholders in EUDAT, including WP5 members, as well as research communities already using EUDAT services and those considering usage of the CDI (including Europeana and EuroFusion).

As a basis for the discussions, in beginning of year two of this phase of the project, we worked out several strategies for integrating object stores into the CDI software stack. They were documented in presentations and discussed with the stakeholders in EUDAT by the task 9.1 members. The following section summarizes the strategies.

3.1.1. Basic Integration Concept

We worked out two basic strategies for integrating object stores into the CDI software stack that implements the EUDAT data management services: the conservative/safe strategy and the brave man's one.

The first of these is a **gradual evolution approach**, which assumes that the object stores are to be integrated at the current B2SAFE back-end. In such a case, the object stores would constitute a storage persistency layer, while iRODS would remain a default user interface and would still implement a rule engine. The rules themselves would be acquired from the DPM (Data Policy Manager). The PID system would also be interfaced using iRODS rules.

While this approach only makes it possible to partially use the power of object stores for data storage and access, it conserves the overall architecture of the CDI and keeps its access interfaces and rule engine untouched. Therefore it is assumed to be easier to adopt into the EUDAT CDI, taking into account the fact that the production services must remain operative and major modifications of their architecture is risky from the sustainability and operations point of view. An overall architecture of the object stores integrated with the current components of the CDI is presented in Figure 1.

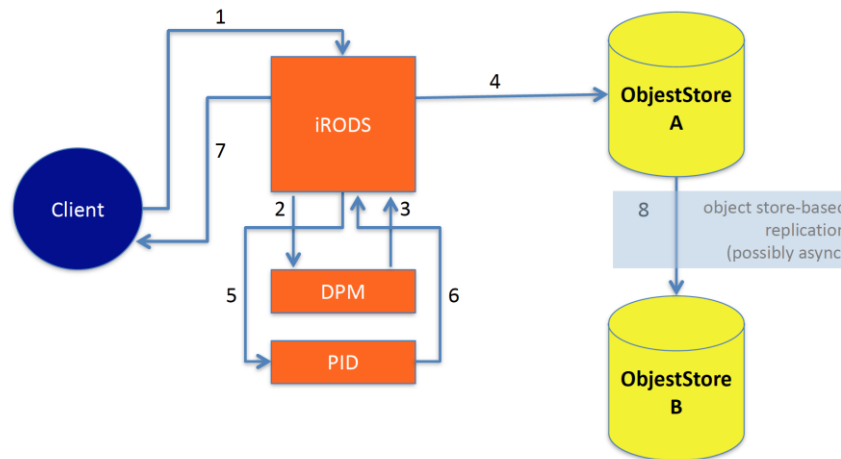


Figure 1: Integration of object stores at the iRODS back-end

Adopting the second, more **revolutionary approach** would require restructuring fundamentals of the EUDAT CDI. This would assume that the B2SAFE interfaces would be replaced with an HTTP-based API. Another assumption would be that part of the data management-related functionality of the EUDAT services will be implemented using object storage internal mechanisms and their extensions. In particular, this would apply to data replication and registration of PIDs. In such an approach, the configuration of the object storage systems' pools would be determined based on the policies defined in the DPM. For instance, rules related to object replication would be translated to an object storage system-level replication configuration. Similarly, policies and rules related to the creation of PIDs and/or their updates would be implemented as extensions to the object stores' logic (for example, using the Python Paste in Swift).

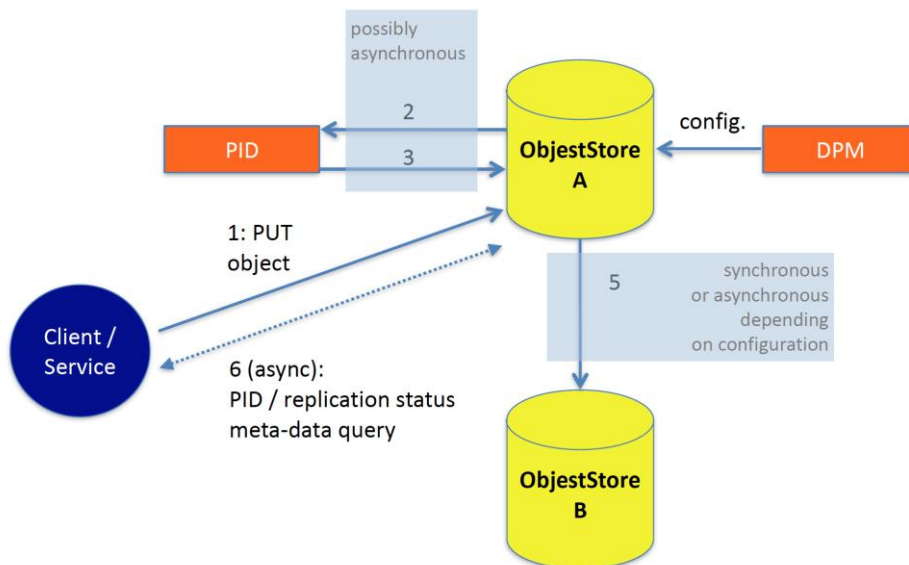


Figure 2: Object storage-based B2SAFE implementation

We documented the integration concepts presented above and presented them to the EUDAT stakeholders for consultancy. Feedback from this process is presented in the following sections of this chapter.

In addition to the conservative and aggressive approaches, we were also considering intermediate solutions, such as enabling the HTTP-based APIs on top of the still-iRODS-based B2SAFE, combined with adopting object stores on the iRODS back-ends. However we concluded that such an approach is overly complex as it requires extensive work on both the iRODS front-end and back-end and yet it still would not bring a real breakthrough.

We also found that potential work on enabling an HTTP-based, S3-compliant API on top of iRODS would be redundant to efforts undertaken by several groups in EUDAT. Information about these efforts came out over the course of the second year of the project with the proposal for HTTP REST APIs for particular EUDAT services. While these works were not targeted at ensuring S3 or Swift protocol support at the iRODS front-end and, as such, were orthogonal to our efforts, we decided to focus on researching a deeper integration of object stores into the CDI services architecture due to the observations mentioned in the previous paragraph – in our opinion enabling S3 or Swift API on top of iRODS will not bring a real scalability and I/O efficiency breakthrough.

3.1.2. Consultancy within WP9

As WP9 is the project activity which provides proposals for the future development of the EUDAT CDI, the JRA was also the natural forum for discussing the concepts related to the integration and exploitation and of object stores.

Consultation with Task Leaders

In addition to the internal T9.1 discussion, the object storage integration concept was presented to other WP9 tasks leaders, including those of tasks T9.2 “Graph-based Data Integration” and T9.3 “Archive Storage Enhancements”. This discussion started in early 2017 and was summarized during the face-to-face meeting of WP9 task leaders in Abingdon, UK. Working out a common view and approach on discussing the adoption of WP9 results within EUDAT was, to a great extent, stimulated and moderated by the new work package leader (Ian Collier) who took over from Shaun de Witt. The brainstorming lead to several conclusions.

First of all, we agreed to present the integrated concept to the Technical Committee (TC) of EUDAT, as well as discussing directions of further work for T9.1, and WP9 in general, with this body. Based on that, Jos van Wezel presented various approaches for adopting object stores as the common EUDAT back-end at the TC meeting on 10-11th May 2016. (One of following sections provides information on feedback from the TC.)

Second, we decided to focus on the integration of 9.1.1 and 9.1.2 by coordinating the efforts made in these two tasks in relation to object stores and HTTP-based federations with the purpose of providing comprehensive data management mechanisms combined with transparent, lightweight and scalable data access solutions (the details are provided in the relevant point below.).

Third, based on the progress reports from the task leaders, the concepts that were presented and the discussions that were held, we identified the opportunity for collaboration between task 9.1 (focused on HTTP-based storage and federations) and task 9.2 (related to examining graph-databases). We decided to explore the possibility of exploiting graph databases and related concepts to deal with metadata storage, management and exploration in the envisioned, future, scalable data storage and management system.

HTTP-based Storage and Federations

Task T9.1.2 “HTTP-based Federation” is complementary to task T9.1.1 “Object Store Evaluation”. Together these two tasks comprise task 9.1 “HTTP-based Storage and Federations”. Therefore particular attention was paid to synchronizing the work in T9.1.1 task with the efforts in the T9.1.2 task, as well as on working out a consistent proposal for integrating object stores and HTTP federation into the future EUDAT CDI. For this purpose, in the second and third quarters of 2016, we organized a series of discussions and brainstorming sessions using video-conferencing and face-to-face meetings involving members of these two tasks.

Based on these discussions, we prepared a concept of an object storage-based B2SAFE service implementation supported by a DynaFed-based federated data access layer. In the integrated concept, object stores provide the basic data storage and access functionality, as well as implement elementary data management mechanisms, including replication and PID creation. As a complement to this, the HTTP-based DynaFed federation deals with transparent access to distributed data replicas, that are created using object storage mechanisms. Lightweight and transparent federation also makes it possible to provide scalability, efficiency and reliability when it comes to data access.

In addition we worked out a method of extending Dynamic Federations by providing support for the S3 protocol on the federation front-end. This would enable fully transparent and unified – from the point of

view of the data storage and access protocol – federation of multiple S3-speaking object storage back-ends. While the DynaFed extension goes beyond the scope of the T9.1.1 task, the task participants were involved with conceptual discussions, including meetings with DynaFed developers at CERN, as well as in the design of the prototype S3 gateway for DynaFed that will provide support for basic S3 queries on top of DynaFed.

Graph Data Bases Integration

The potential of graph database technology to provide a future-proof solution for dealing with metadata and representing the relationships between digital objects within the EUDAT CDI had already been identified in the first year of the project⁹³. In deliverable D9.1 we pointed out and envisioned several possibilities for applying graph-databases in the EUDAT CDI data management services, including handling file system-like hierarchical relationships between objects, representing complex relationships, for example, between objects and users (ownership) or data objects and data sources (provenance), as well as implementing efficient search algorithms.

During the meeting in May 2016 and the follow-up discussions, we defined concrete aspects where graph database technology can be applied in the T9.1.1 context. As the result of these discussions, a concept of integrating object stores and graph database solutions has been worked out. The task participants have implemented the prototype of the integrated system. It included the object storage-based B2SAFE-like data storage and access service. It also encompassed mechanisms for metadata handling, exploration, and searching, as well as metadata-based access supported by graph databases. The details of the proof of concept are provided in a paper⁹⁴.

3.1.3. TC Feedback

A summary of the object storage evaluation work conducted in the first year of the project was presented at the Technical Committee meeting on the 10-11th May 2016, by Jos van Wezel, on behalf of WP9. In general, the feedback from the TC about the vision that was presented was positive with several comments being given regarding its possible adoption. Overall, in the opinion of the TC, object stores may constitute a basis for a data persistency layer of the future CDI infrastructure, assuming that other components will also be restructured. It was also noted that adopting the aggressive path of integration of object stores into CDI is theoretically possible, assuming that the necessary adaptation work is undertaken on the DPM side and in PID components.

The positive feedback from the TC related to possibly integrating object stores into the CDI provided motivation for further technical discussion with participants from WP5, the Service Activity of the EUDAT project responsible for scouting and integrating new technologies into the CDI, as well as for architecture design of the future CDI within WP9.1.

3.1.4. Feedback from Service Activities

The overall project work plan was organized assuming that it would be possible for the service activities to adopt the results of WP9 in order to improve the scalability, robustness and quality of the EUDAT services, as well as extending their functionality. Another assumption that was made was that JRA provides prototypes of the solutions and/or their components, as well as far-looking architectural visions and system development recommendations.

Following these assumptions, we presented the vision worked out in WP9.1 to the WP5 leader and the service development leaders, in order to exchange ideas and opinions and to receive feedback. In the following section we summarize this work.

⁹³ J.Rybicki, V.Bunakov, P.O. Meo, S.Kindermann, A.Queralt. Graph-based Data Integration in EUDAT Data Infrastructure. ALLDATA 2016: The Second International Conference on Big Data, Small Data, Linked Data and Open Data. February, 2016. https://www.thinkmind.org/download_full.php?instance=ALLDATA+2016

⁹⁴ St. Vieth, B. (Corresponding author)* ; Rybicki, J.* ; Brzeźniak, M. Towards Flexible Open Data Management Solutions. Croatian Society for Information and Communication Technology, Electronics and Microelectronics MIPRO Rijeka, Croatia, 2017. ISBN: 978-953-233-090-8

Consultancy with WP5 Leader

In the third quarter of 2016, we consulted the WP5 leader, Mark van Sanden, regarding the possibility of integrating object stores into the EUDAT CDI as a common back-end and data persistency layer for several services. We also presented the possibility of implementing the B2SAFE-like service based on object stores. In particular we presented an idea for a **B2SAFE-light service** that implements functionality similar to B2SAFE, including replication based on internal mechanisms of object storage systems and PID creation integrated using object storage extensions. We have also discussed combining this approach with the dynamic, HTTP-based federation for data access.

The overall feedback was positive. The potential of scalable object stores and the possible benefits for the EUDAT CDI were noted. However it was also concluded that taking the brave integration path might require substantial adaptations of existing EUDAT services, including B2SAFE, B2SHARE, B2DROP as well as DPM. In this context, we were advised to consult with the teams responsible for those particular services.

We also agreed on the idea of providing an alternative implementation of the B2SAFE service, **B2SAFE-light**, solely based on new technologies, including object stores, dynamic HTTP federation and graph databases. With such an approach, B2SAFE-light might be used within the EUDAT CDI with the users and use-cases that cannot be effectively served by traditional storage systems, for example, due to a high expected I/O rate or a vast volume of data that needs to be stored.

During the discussions, the WP5 leader also pointed out that, in the context of the EUDAT CDI requirements, cross-technology data replication has to be considered. Similarly, approaches for federating object storage clusters across management domains must be worked out. According to the WP5 leader, these are among the crucial issues that will be faced while trying to deploy B2SAFE-light within the distributed infrastructure of the CDI. Due to scale of the EUDAT CDI, and the fact that it spans infrastructures belonging to various organizations, it may include storage clusters implemented using different object storage technologies (for example, Swift vs Ceph) or may not allow tight integration of object store clusters with remote installations due to administration policies.

Based on these suggestions, we analysed several ways of integrating technologically heterogeneous and organizationally independent object storage systems within the T9.1 team. Among the options that were considered was the idea of providing object storage system extensions making it possible to replicate data to another object storage system. Such a solution could, for example, be based on the module for Swift performing PID registration and data replication to separate or remote Swift or Ceph clusters, implemented for Swift using the Python Paste framework. We also investigated the possibility of implementing an object storage technology-agnostic solution that would act as the intermediate party while implementing data management processes on top of a distributed object store. For instance, data replication could be implemented in a similar way as FTS-realised third party transfers between storage systems.

Based on the feedback from the WP5 leader, as well as broader consultancy in the project and beyond (see following sections), we came to the conclusion that implementing the data management mechanisms on top of loosely coupled object stores is the best way to attain the desired scale and level of modularity and cross-technology compatibility of the data persistency layer for the future EUDAT CDI. We also concluded that further research on such concepts should be coordinated with the efforts undertaken in T9.1.2 related to federation mechanisms based on HTTP, aiming, among other things, at enabling support of the S3 protocol at the DynaFed front-end.

Consultancy with Service Teams

In the third and fourth quarters of 2016 we contacted the B2SAFE and B2SHARE teams in order to consider our vision in the light of their opinions and the development roadmaps for the service, as well as looking at possible adoptions of our results.

B2SAFE

The task T9.1.1 and T9.1.2 leaders presented the concept of a **B2SAFE-light** service based on integrated object storage systems and an HTTP-based access federation layer to the B2SAFE team (including Claudio Cacciari) during the third technical meeting of this phase of EUDAT in Bologna. The feedback was as follows.

B2SAFE-light is meant to be an alternative implementation of basic B2SAFE functionality, including data replication and registering PIDs, but based on object stores with extensions.

First, the B2SAFE team sees the potential of applying scalable and fully distributed technologies to the EUDAT CDI. This applies to the data storage and access components, as well as data management and meta-data handling.

Second, the B2SAFE team shares the view of the T9.1.1 participants that the most profitable and promising way of exploiting object stores is to develop a service that will provide an alternative implementation of the service functionality based entirely on modern technologies. There is agreement that attaching object stores at the back-end of iRODS does not make it possible to use the full power of object stores and HTTP federations. The B2SAFE developers also see the benefits of applying modern databases technologies, such as graph databases and/or no-SQL databases, to handle the metadata of the objects and represent their relationships.

Third, the B2SAFE team agreed that there are use-cases that demonstrate the necessity of applying new technologies due to the capacity required of the storage service, the expected I/O rate and I/O parallelism, as well as the data ingest or data access performance. The latter applies both to the expected throughput with uploads and downloads of massive objects, as well as to the number data object ingest or access operations supported per second. In this context, it was suggested that we produce a proof of concept of our proposal by implementing the prototype and performing its experimental evaluation. It was also advised that the prototype benchmarks should use the data sets that are representative for use cases and user communities that have requirements reaching beyond the current capabilities of the EUDAT CDI.

In this context, we were also advised on the positioning of our conceptual B2SAFE-light service within the EUDAT services portfolio. In particular, in the opinion of the B2SAFE team, the relation to the existing B2SAFE implementation should be clarified. We agreed on the view that B2SAFE-light is, in fact, another alternative service that provides B2SAFE-like functionality and may be run in the CDI data centres in parallel to B2SAFE for the benefit of those user communities whose requirements cannot be served with current implementations. Assuming the different target group of users and user community-level services to be integrated with B2SAFE-light, its interactions with other EUDAT services can be limited. This will make it possible to simplify its functionality and optimize the implementation, thereby allowing it to address high-throughput use cases. As a side effect, complying to standards at the access layer (for example, mimicking S3 or Swift) should also be possible.

Fourth, the B2SAFE team pointed that the CDI solutions that are being envisioned should take into account the latest Data Policy Manager developments. Overall, the DPM-related work is aimed at enabling user communities to define policies related to data management on a relatively high-level of abstraction, in a technology-agnostic way. For instance, data durability will be the target measure to be agreed with end-users, rather than the number of replicas in filesystems or object stores or their exact locations. In the context of applying object stores to the CDI, these DPM-level policies could be translated into storage system-specific configuration settings.

Fifth, the B2SAFE team leader expressed the opinion that, while enabling the EUDAT CDI software stack to interface to S3-speaking storage would be an important step in improving the CDI's support for industry standard data access and management protocols, such development would not replace a holistic re-design of the EUDAT CDI that would target better integration and scaling of the EUDAT services than is currently possible. In fact, harmonization of the data models considered by the EUDAT services, including B2SHARE and B2SAFE, should be performed in addition, or as a complementary action, to back-end storage protocol

unification. Moreover, redesign of the service architectures might be required in order to remove any possible centralization points and bottle-necks that may arise while dealing with future storage needs.

Overall the discussion with the B2SHARE team brought several important factors into our future architecture design work and prototype activities. First of all, it was recommended that we decouple our concept from the current B2SAFE implementation, which will make it possible to avoid its centralization points. Such a strategy is central to our proposals. Second, it was suggested that we position the conceptual B2SAFE-light service as an independent solution to B2SAFE, which confirms our beliefs that integrating object stores with iRODS is not a promising way to ensure real progress compared to the current status quo. Adopting a fresh start approach to providing object-storage based services within the EUDAT CDI also guarantees freedom while selecting access protocols. S3 and Swift are natural candidates in this area as they are recognized standards on one hand, and they allow custom extensions on the other (for example, those needed to store, maintain or present specific metadata fields such as those including PID information). Third, it was underlined that current work within the B2SAFE team related to DPM development is aimed at abstracting policies to a user-understandable level that is independent of the actual configuration of the infrastructure (for example, IP addresses of the storage endpoints, or storage system technologies). While this work is still at a preliminary stage (as the current DPM implementation is still conceptually bound to iRODS rules), the strategic direction of this work is in line with our efforts. The fact that the DPM is envisioned as the ultimate source of data management policy information in the EUDAT CDI should be taken into account while conducting our design (and potentially prototyping) work.

HTTP-API

Several groups within the EUDAT CDI work on enabling HTTP APIs for EUDAT services, including B2SAFE, B2STAGE and B2SHARE. Therefore, during the meetings in 2016, we attempted to align our activities with these efforts. Based on the consultations that were conducted, we concluded that our T9.1 work on enabling HTTP access to storage systems is to be considered as independent of the work regarding the front-end HTTP APIs.

First of all, from the architecture point of view, the storage back-ends are positioned relatively low in the layers of the CDI software stack, compared to the HTTP APIs of B2SAFE, B2STAGE and B2SHARE. Our work applies to the low-level data storage, access and management services. In contrast, the HTTP API group work is targeted at providing ‘wrappers’ for the high-level functionality of the existing services. For instance, the B2STAGE API provides the possibility to steer data movement between EUDAT and external storage systems through an HTTP API while actual transfers are performed through protocols supported at various end-points including iRODS and GridFTP. The B2SHARE API is used as an abstraction layer of B2SHARE services which is used in order to build the B2SHARE service portal and perform other possible integrations. Finally, the B2SAFE API makes it possible to deposit data objects, list deposits and query the metadata included into the PID record of a specific file, however, while it is improving the current situation by enabling HTTP-based access, it does not address the scalability and performance issues of the underlying B2SAFE services’ implementation.

Moreover, there is not yet an integrated API for all the EUDAT services, as their development started, and was motivated, by the needs of various services (and user groups). While the effort on converging these APIs into an integrated protocol or API suite is ongoing, this work has not been finished.

Based on the analysis we conducted, we decided to keep the focus of T9.1 on envisioning and prototyping scalable implementations of basic data storage and access functionalities, as well as fundamental data management mechanisms, available through standard HTTP-based protocols like S3, with minimum possible extensions.

3.2. Feedback from Outside the Consortium

Increased usability and adoption of the EUDAT CDI, and its services, is at the heart of the EUDAT2020 project, including WP9. Therefore we took the feedback from the external stakeholders into account in our conceptual and design work related to adopting object stores into the EUDAT CDI.

3.2.1. EC Reviewers

The EC experts in the first year's EC review provided high-level guidance related to the EUDAT CDI design and the development of the EUDAT services. They recommended that EUDAT should "develop a more modular architecture to enable a wider range of infrastructure services to be integrated". They also expressed an expectation that EUDAT will "keep up with the latest technical innovations" and that "modular and flexible architecture were to be put in place". They also suggested that we should "integrate de facto standards and all forms of infrastructure service offerings." In addition, the importance of enabling HTTP-based access and federation was noted: "HTTP federation is to be explored as a transport layer for the EUDAT bus. This will provide a real breakthrough if achieved."

These comments are in line with the direction of the work adopted for our task. First, we evaluated the technological components, such as object stores that support industry standards including S3 and Swift. Our research confirms that these technologies are promising candidates for building the future EUDAT CDI. Our work also showed that they make it possible to use a modular approach to various aspects of data management. In particular, data storage and access, as well as basic data management tasks, can be encompassed in the object storage systems and access federation layer, while advanced data management and metadata handling can be realized by the higher level functionality and services. Moreover, integrating the object storage and access federation solutions into the CDI both requires and enforces supporting standard protocols, such as HTTP and/or S3 on the back-end of high-level EUDAT services such as B2SHARE, B2SAFE, B2STAGE and B2DROP.

Second, within our work we considered technological improvements in the area of object stores and HTTP access federations, as well as the benefits that their integration could bring to the EUDAT CDI. These include increased scalability, full decentralization of data storage and access, as well as improved cost efficiency.

Additional positive feedback related to the concepts that were proposed was received during the second interim EC review. The EC experts admitted that "object stores offer huge advantages for EUDAT, especially when deployed over distributed architectures". They also agreed that overall, adopting HTTP-based storage enhances the possibilities for modularizing the architecture of the services.

While this feedback is in line with the previous comments received in 2015, a new aspect that it brings to light is that the reviewers perceive fully distributed deployments of the object stores to be the most beneficial and promising approach to exploiting this technology. This view is consistent with our discussions and analysis, however, stated so clearly, it provides additional motivation for the efforts we made in the second year. Actually, the scope of our work in this period reached beyond the usage of local instances of object stores and extended towards aspects related to the integration of object stores in fully distributed setups, including scalability, reliability, performance of the overall infrastructure and compatibility with the existing software stack. We also analysed and proposed several strategies for object storage adoption and adaptation of existing systems. Finally, we also proposed the use of HTTP federations for exploiting multiple heterogeneous storage systems and their instances, thereby offering multiple access endpoints

Apart from the positive feedback on the general architectural proposals of our task, the EC experts raised several concerns related to a consistency model in object stores. In the first year, the experts commented that: "Those working in this WP are encouraged to remember the essential nature of transactional integrity. Unless there is a robust DBMS on top of the data store/repository, it is not possible to ensure/enforce ACID compliance." In the second year they pointed out that: "They [object stores] maintain little transactional integrity, so data operations that involve updates after processing have to be managed manually". In the first year they also highlighted several limits of the object store technology, including issues with their direct

applicability as the storage back-ends for HPC processing or Big Data-in-motion systems (which are however not the central use cases for the B2SAFE, B2SHARE or B2DROP services).

It is important to note that, while raising the issues related to the data integrity model in the context of the proposed adoption of object stores in the EUDAT CDI, the reviewers noted a more general and fundamental challenge related to future EUDAT CDI development. In fact, enabling the EUDAT CDI to exploit the real potential of object stores, as well as other modern, fully distributed technologies like HTTP-based access federations and graph or noSQL databases, might require holistic re-thinking of the assumptions related to data and metadata models across the CDI. In fact it is a part of our long-term vision to analyse the impact of adopting fully decentralized data storage access, as well as data and meta-data management components, into the overall architecture of the CDI.

3.2.2. External Data Management and Cloud Services Experts

While aiming to make our vision be aligned with the broader technology developments and trends in current R&D, we consulted stakeholders from outside the EUDAT project, including data management and cloud services experts at institutions and projects dealing with large-scale data management such as CERN, NRENs and Geant.

Overall, according to the opinions that were gathered, the integration of object stores into data management projects, solutions, services and products is a common strategy than brings significant benefits.

This applies to WLCG, and other large-scale projects conducted by CERN and its partners, amongst others. On the infrastructure side, CERN tests and deploys Ceph-based object storage systems at the multi-petabyte scale^{95, 96} in addition to its EOS platform⁹⁷. On the software development side, recent extensions of DPM and FTS were intended to enable HTTP-based data storage, access, and transmission, as well as supporting the S3 protocol itself. DAVIX, a high-performance HTTP client developed by CERN, which is widely used in Grid and cloud projects, currently includes several S3-oriented optimizations. DynaFed, another solution from CERN, was recently equipped with S3 and Microsoft Azure back-end plug-ins, in addition to WebDAV. The new version of the file transfer service, FTS3, supports HTTP and S3, allowing for third-party transfers to and from the WebDAV sites, as well as transfers between S3-speaking endpoints⁹⁸. According to the experts at CERN that we consulted, it is a strategic direction at CERN, and in WLCG and related projects, to adopt object stores in order to exploit their architectural and technological potential for dealing with large-scale data volumes in an effective, reliable and cost-efficient way. While it might require redesign of significant parts of the services' software stack, this effort is considered profitable in the long-term and at the large scale.

R&D projects in the area of cloud and grid computing and storage are also making progress in the adoption of object stores in parallel to traditional storage. Amongst others, INDIGO-DataCloud conducted a dedicated task on integrating object stores at the data centre level, as well as at the federated data management and cloud orchestration layer⁹⁹. Also the EGI-ENGAGE project enabled DataHub, its data federation component, to support object stores exposing S3 or Ceph protocols¹⁰⁰ in addition to other storage back-ends. These investments were motivated by the belief of the system architects working in these projects, and in the EGI organization, that object stores will be an important part of the infrastructure landscape in coming years.

In parallel, data management specialists at NRENs across Europe and globally (contacted through various fora, including TF-Storage of TERENA/Geant, the OpenStack operators group and the GN3/4 project collaborations) deploy object storage systems as the back-ends of their data management, content serving and IaaS, PaaS and SaaS cloud services. According to our contacts and surveys, Ceph is the most widely adopted as the back-end for OpenStack-based private services, as well as the object storage service. Swift is

⁹⁵ <https://cds.cern.ch/record/2015206/files/CephScaleTestMarch2015.pdf>

⁹⁶ <https://www.openstack.org/videos/vancouver-2015/ceph-at-cern-a-year-in-the-life-of-a-petabyte-scale-block-storage-service>

⁹⁷ <http://information-technology.web.cern.ch/services/eos-service>

⁹⁸ <https://indico.cern.ch/event/394833/contributions/2296031/attachments/1335535/2008698/preGDB20160913-S3.pdf>

⁹⁹ D.Salomoni et al. INDIGO-Datacloud: foundations and architectural description of a Platform as a Service oriented to scientific computing. <https://arxiv.org/pdf/1603.09536.pdf>

¹⁰⁰ <https://www.egi.eu/about/newsletters/introducing-the-egi-datahub-prototype/>

also considered to be a reliable object storage solution, which is especially suitable for geographically distributed installations. In addition to efforts related to operating on-premise object stores, a public cloud services brokerage facility developed by GN4 includes S3-enabled storage services from commercial providers, amongst other offerings. Also experts at larger universities and research institutes we contacted are working on, and envision further adoption of, object storage systems in private cloud installations and the growing consumption of public, hybrid and community-cloud-based S3 services, especially for content handling and serving, backup/archive and long-term storage.

Overall, the interaction with the external experts confirmed our beliefs related to the suitability of object storage systems for large scale, long-term, scalable and reliable data management systems. Another lesson from these discussions is that EUDAT should take into account, and can benefit from, the growing demand for scalable and cost-effective object storage systems, while trying to target the new R&D communities.

3.3. Interactions with User Communities

We presented the new opportunities and features of the envisioned, fully scalable, object storage-based services to the EUDAT user communities. These contacts involved stakeholders already using or piloting usage of EUDAT services and those not yet involved in usage of EUDAT CDI. In this section we present the results of the consultation with the Europeana association and the fusion community, as we find the data management challenges these communities face to be representative of those of many of the large-scale projects and R&D initiatives.

Overall, the discussions we conducted confirmed that the growing needs regarding the volume of data to be stored, preserved, accessed and made referable, as well as the complexity of the structure of the data in terms of the number of objects, their relationships and the expected data disposal and access rates (bandwidth and objects per second) require applying new, fully scalable, distributed and decentralized storage technologies.

3.3.1. Europeana

In the case of the Europeana association, which spans 3500+ digital heritage and memory institutions, several efforts have already been undertaken with the aim of providing a common, future-proof solution for implementing data and metadata management functionality for the benefit of cultural institutions. These included developing the Europeana Shared Services^{101,102,103} within the Europeana Cloud project¹⁰⁴.

Although Europeana's focus is on metadata management, including its flagship service which aggregates the metadata from multiple sources and make it possible to access, explore and exploit the data through a portal and APIs, however they are considering providing additional services, including content storage, access, delivery, presentation and processing. Notably, Europeana already provides services for hosting and delivering selected datasets, such as the Europeana Newspapers collection. Interestingly, this solution is based on the cloud infrastructure that streams content from OpenStack Swift at PSNC directly to the European Library portal¹⁰⁵. However, with above mentioned exception, Europeana is still on its way to develop and provide content storage, access, delivery, presentation and processing solutions at large scale.

Overall, according to the Europeana experts we contacted during the EUDAT-Europeana collaboration in the context of our so-called data pilots, the scale of adoption of cloud services for data storage in cultural institutions is still low, while the demand for data and metadata management solutions is huge.

Currently, Europeana deals with over 50 million digital objects that are typically comprised of several high resolution images each (resulting in an average of 1GB per digital object). Taking into account that, according to estimations by the Europeana experts, only 10% of the cultural heritage assets have been digitized so far,

¹⁰¹ <https://confluence.man.poznan.pl/community/display/ECLLOUD/Europeana+Cloud+User+Documentation>

¹⁰² <https://github.com/europeana/Europeana-Cloud>

¹⁰³ <https://cloud.europeana.eu/docs/>

¹⁰⁴ <http://pro.europeana.eu/blogpost/europeana-cloud-shared-infrastructure-for-european-cultural-cont>

¹⁰⁵ <http://www.theeuropeanlibrary.org/tel4/newspapers/issue/3000117723548>

the potential volume and complexity of the target datasets is enormous: in the range of hundreds of PBs and billions of data objects. In addition, the massive character of the Europeana audience, as well as the need to conduct research on these datasets in many disciplines (including history, linguistics, anthropology, and social sciences), results in high expectations related to the rate, latency and parallelism of the data and metadata access operations that have to be served by the future cloud and storage platform backing Europeana.

This creates a great potential use-case for infrastructures such as the EUDAT CDI and an interesting challenge. We believe that (and the Europeana experts we consulted share this view) the Europeana use-case can be best approached using scalable systems, among which object storage may play important role. Object stores with their fully distributed architecture, and eventual support for data consistency models, are suitable for holding Europeana datasets, as they are massive, rarely changing, digital assets. When combined with HTTP-based access federation, and application and portal-level services employing proper architecture, object stores can make it possible to have effective and parallel access to Europeana datasets from many geographical locations.

3.3.2. Fusion

The fusion community¹⁰⁶ is also looking for future-proof solutions for large-scale storage, as well as sharing and processing large and complex datasets. These would enable the numerous fusion research groups, including plasma physicists, mechanical engineers, materials scientists and advanced robotics researchers, to greatly improve the scope, scale and efficiency of their collaborations. These groups remain relatively fragmented and the reuse and sharing of data is limited, while making significant research progress in the fusion domain requires more intensive data-driven collaboration.

Integrating and coordinating fusion data-based research faces several interesting issues. First of all is the overall volume of data. According to fusion experts, ITER¹⁰⁷ itself will be generating 2PB of raw data per day, which is more than all the existing European tokamaks. With the addition of datasets created by simulation and modeling, the rate of data deposit and access will grow dramatically once ITER becomes operational.

Another challenge is the demand for global data distribution, data sharing among groups, and the need to ensure data is available at high performance in the locality of the large-scale processing resources, for example, HPC centers such as CEA, CINECA, FZJ, STFC and PSNC and those beyond Europe. According to the experts we consulted, although the fusion community has a relatively low level of experience with federated data management, transparent cross-location data access is in fact essential for high-performance, parallel and distributed analysis of fusion datasets.

Interestingly, CCFE, a partner within the Eurofusion project, has already started exploring the possibilities of using cloud systems for bursting computing and data processing beyond the capabilities available in its data centres. This has created a basis for adopting external resources, and increased the awareness of the fusion community in relation to the opportunities and challenges associated with cloud computing and storage. Several interactions and consultations were also taking place between PSNC, STFC, the T.9.1.1 task participants and fusion partners (including CCFE and ITER), related to re-designing the future data and metadata management platform for ITER. The options being analysed included the potential usage of scalable storage systems, such as object stores, along with access federations (the PoC was planned based on Ceph clusters at PSNC and STFC, combined with DynaFed) and the possible adoption of flexible metadata storage, access, exploration and exploitation mechanisms based, for example, on graph databases.

Partially as a result of these interactions, a proposal for a centre of competence within an integrated project which is part of the EINFRA-12 call, has been prepared and accepted for the final project proposal (that passed EC evaluation in 2017). Part of the work in this joint initiative between EUDAT, EGI and Indigo-DataCloud will be to extend the unified data access (UDA) component of the fusion software stack by

¹⁰⁶ <https://www.euro-fusion.org/>

¹⁰⁷ <https://www.iter.org>

enabling it to exploit multiple data copies created by EUDAT's B2SAFE service and allow fast data access based on EUDAT's B2STAGE.

Overall, the fusion experts perceive geographically distributed object stores to be a promising approach for dealing with the storage of large-scale datasets, and HTTP-based federation to be a suitable future approach for providing unified and efficient access to data. While exploiting the potential of these technologies would require adopting the fusion data model as well as system architecture re-design and software implementation changes, the partners in the fusion community share our beliefs about the profitability of such investments in the long-term.

3.4. Summary of Discussion Outcomes

In the previous points we discussed the consultancy process conducted by the T9.1.1 participants related to the concepts and ideas proposed in the first year of the project. We also presented and analysed comments and feedback gathered through this process from various sources, including EUDAT stakeholders as well as external experts.

In general the discussions show that our proposals and concepts are convergent with the interests, research and directions, as well as the future development plans, of groups within EUDAT. The need for adopting fully distributed systems for large-scale data management has also been noticed in other R&D projects and across the community of experts responsible for designing academic and research infrastructures.

Overall the outcome of the discussions provided the motivation for further research on adopting object stores for large-scale, long-term storage services. In addition, the requirements and expectations that were articulated have been taken into account while performing and planning proof of concepts and designing the work documented in sections 4 and 5 of this document.

It is important to note that, while we noticed a general interest in exploiting the potential of object stores for reliable, long-term storage of vast volumes of data, we also predict that the actual adoption of object stores within the EUDAT portfolio will be dependent on the deployment of object storage systems at partner sites. We are aware that until object stores become ubiquitous, the adaptation of the existing mechanisms and services in EUDAT will occur relatively slowly. However, based on the trends observed in the storage infrastructures, we assume that the results of our JRA activity will be applicable in the near future.

4. EXTENDED PROOF OF CONCEPT

In the second year of the project we conducted an extended proof of concept of the ideas proposed in the first year. The prototype design and implementation also took into account the analysis reported in section 2 of this deliverable, as well as the discussions and consultancy reported in section 3.

The prototype materializes the second, more revolutionary, approach to the integration of object stores into EUDAT CDI as discussed in section 3.1.1 of this deliverable. This approach assumes that the interfaces of B2SAFE are replaced with an HTTP-based API and that part of the data management related functionality of the B2SAFE service is implemented using object storage internal mechanisms and their extensions.

In particular, in our prototype, the OpenStack Swift object storage system acts as the data storage and access layer. It also implements elements of data management. This includes data replication based on container synchronisation mechanisms. In addition, PIDs are created based on the integration of EPIC, the persistent identification system used in EUDAT. EPIC functionality is integrated within the OpenStack Swift I/O request processing chain. Our prototype also uses the graph database Neo4J¹⁰⁸ which stores metadata and provides metadata search functionality that makes it possible to explore the data stored in the system using a flexible mechanism for metadata queries based on Cypher, the Neo4J query language¹⁰⁹.

The prototype provides an alternative implementation of the B2SAFE service. As discussed in section 3.4, such an implementation of basic B2SAFE-like functionality could be adopted in the EUDAT portfolio as the B2SAFE-light service, which would use object stores and graph databases, as opposed to B2SAFE which is based on iRODS.

The details of this prototype are described in a paper¹¹⁰. In this section we provide an analysis of the prototype design and implementation, as well as comments on the results of the evaluation work.

4.1. Prototype Design

The prototype design assumes the implementation of functionality similar to B2SAFE, that is, data replication and PID registration for digital objects stored in the service which is based solely on the functionality of the object storage systems and their extensions, with no additional layer on top.

In our prototype implementation, all the basic aspects of data management considered as integral to the EUDAT B2SAFE service are implemented using OpenStack Swift. This includes data storage and access, as well as ensuring the required level of data redundancy and integrity control and protection. In the prototype, the OpenStack Swift's built-in mechanism for container-to-container replication¹¹¹ is used in order to perform data replication across containers stored in two distinct, administratively independent, clusters. Swift integrity checking mechanisms also periodically check for the consistency of the containers, and take corrective action if needed, for example, by re-creating missing or corrupted copies of the data objects.

An important feature of the prototyped implementation of the B2SAFE-like functionality is that OpenStack Swift makes it possible to guarantee the required level of data redundancy by using its built-in mechanisms, according to the declarative configuration settings of the storage pool. On the contrary, in the case of the iRODS-based B2SAFE, replication policies have to be explicitly implemented using iRODS rules. A step in the direction of abstracting policies from the implementation has been made by EUDAT's B2SAFE group while developing the DPM, which makes it possible to define high-level features related to data management, such

¹⁰⁸ J. Webber, "A programmatic introduction to Neo4j," in SPLASH '12: 3rd ACM Annual Conference on Systems, Programming, and Applications: Software for Humanity, Oct. 2012, pp. 217–218

¹⁰⁹ <https://neo4j.com/developer/cypher-query-language/>

¹¹⁰ St. Vieth, B.; Rybicki, J.*; Brzeźniak, M. Towards Flexible Open Data Management Solutions. Croatian Society for Information and Communication Technology, Electronics and Microelectronics MIPRO Rijeka, Croatia, 2017. ISBN: 978-953-233-090-8. http://docs.mipro-proceedings.com/dcviz/dcviz_11_4110.pdf

¹¹¹ https://docs.openstack.org/swift/latest/overview_container_sync.html

as the data redundancy level, and translate them automatically into iRODS rules. However, such an approach involving the DPM and iRODS is still in an early phase, and is conceptually bound to the deployment and configuration details of the iRODS instances.

In addition to replication, the Swift storage cluster calls the remote EPIC PID system for persistent identifier creation for every deposited object, based on the extension implemented using Python Paste framework¹¹².

An interesting feature of Swift is the fact that extra functionality can be added to it using relatively easy and open interface. Swift uses the Python Paste framework in order to build the processing pipeline for I/O requests. Thanks to this, it is possible to extend the standard system functionality by implementing additional middleware and including it in the pipeline. The following example illustrates a fragment of the configuration file that depicts a standard Swift processing pipeline extended with the `persistent_identifier` middleware that creates PIDs in the EPIC system.

```
[pipeline:main]
pipeline = authtoken keystoneauth persistent_identifier cdmi proxy-server
```

It is important to note that plugins developed using Paste are well integrated with the general Swift system logic and mechanics. This allows great scalability of the combined solution, as the extensions can benefit from the internal system parallelism and the mechanisms supporting asynchronous communication with internal and external processes and services.

Last, but not least, it is noticeable that the prototype design assumed its operation across management domains, which is one of the fundamental requirements in the EUDAT CDI. Swift's container-container synchronisation makes it possible to copy objects between administratively independent clusters that belong to the same realm, defined for synchronisation purposes. This mode of synchronisation does not require the tight integration of clusters that is typically not supported in distributed environments such as EUDAT.

Providing an OpenStack Swift-based implementation of B2SAFE-like functionality (that can be deployed in a cross-domain mode) is an important step towards achieving object storage system federation for the EUDAT CDI. Conceptually a similar setup (in the aspect of data replication) can be implemented based on Ceph, by applying multi-site configuration for the RADOS gateway^{113,114}.

However it is also important to note that these approaches allow only clusters implemented in the same technology to perform automated data replication. Therefore, further work would be required to enable cross-technology implementation of B2SAFE-like functionality based on heterogeneous object stores.

4.2. Implementation and Evaluation

Achieving B2SAFE-like functionality for the prototype based on OpenStack Swift necessitated developing a custom Swift extension by providing so-called middleware for the Python Paste framework-based request processing pipeline. Similarly, iRODS rules were developed by the EUDAT partners in order to implement replication and to integrate PIDs into B2SAFE workflows. Aside from the architectural advantages of OpenStack's approach to extending functionality (discussed in the previous point), the technologies that were used in Swift have several strengths relating to practical aspects of the implementation of the extensions.

First, Swift extensions can be implemented in Python, which is one of the standard programming languages and is widely adopted in IT, in contrast to the iRODS rules language. Thanks to this, providing Swift extensions does not require custom programming skills and knowledge specific to iRODS rules. One may also assume that the implementation of efficient and reliable data management logic in one of the popular programming

¹¹² <https://pypi.python.org/pypi/Paste>

¹¹³ <http://ceph.com/wp-content/uploads/2017/01/Understanding-a-Multi-Site-Ceph-Gateway-Installation-170119.pdf>

¹¹⁴ <http://docs.ceph.com/docs/master/radosgw/multisite/>

languages is easier to do, and less error-prone for Python than would be the case with iRODS rules. In fact, feedback gathered from the developers within EUDAT, including those involved in the discussed prototype integration, confirms this view. While new version of iRODS provide plugins for writing rules in Python^{115, 116}, the actual usability of this solution has not been tested in EUDAT to the best of our knowledge.

Second, the Python Paste framework introduces relatively little overhead as far as I/O processing in Swift. This is confirmed by the prototype evaluation which shows that the overhead of the simple OpenStack plugin constitutes a small percentage of the total I/O request processing time needed to store data objects in the Swift system¹¹⁷. Evaluation of the prototype also demonstrated that additional latency caused by the plugin does not degrade the aggregated throughput achieved while ingesting data to the object store. This is possible despite the fact that the per-request delay can be noticeable, since scalability and high parallelism of I/O processing in object stores makes it possible to achieve overall high performance if multi-threaded data upload and retrieval is conducted.

4.3. Conclusions from the Proof of Concept

The proof of concept conducted by Task 9.1.1 participants provided a possibility to practically verify elements of the concepts and ideas related to implementing EUDAT services using object stores, and fully distributed, decentralized components, in general.

The prototype design and implementation demonstrated that scalable architecture of object stores, and their modular implementations in such a way as to enable functionality extensions, make it possible to build the data storage, data access and long-term data management workflows required for EUDAT in an efficient and reliable way.

It has also shown that implementing the concept of a B2SHARE-light service is already realistic at the current stage of the development of object storage systems and federation solutions. While cross-technology replication is not yet possible off the shelf, replicating data across administrative domains is possible.

¹¹⁵ https://docs.irods.org/4.2.1/plugins/pluggable_rule_engine/

¹¹⁶ https://github.com/irods/irods_rule_engine_plugin_python

¹¹⁷ The prototype evaluation presented in the referenced paper shows that the extra delay related to integration of Swift with Kafka-based transaction log fits within 0,16 seconds i.e. <10% of the total upload time for the files of the size in range of 100MB.

5. FROM OBJECT STORES INTEGRATION TO FUTURE ARCHITECTURE OF EUDAT

The assumption of T9.1.1 and the WP9 JRA overall was to “strike a balance between ambitious, far-reaching goals and direct applicability of the results with immediate relevance of the solutions” (DoA).

As a complement to the work presented in the previous section of this document, and partially based on its results, we envisioned a future architecture of EUDAT, which assumes the adoption of modern, fully distributed data storage and management technologies. This include, amongst other technologies, object stores applied for providing low-level data storage and access, as well as data persistency services.

In this section we present the resulting high-level vision of the future CDI, and provide views on potential future activities of EUDAT2020 in the third year of the project related to our concepts.

5.1. Scene for Future Architecture of EUDAT

During the second year of this phase of the EUDAT project, we analyzed advances in object stores technologies and their adoption, as well as market trends related to their usage. This work showed that object stores are a promising, mature and widely adopted technology suitable for building reliable, high performance and cost efficient data storage, access and management systems.

We have also discussed the concepts related to applying object stores as the building blocks for data persistency services for the CDI within EUDAT and beyond. The feedback from these consultations was that there is an understanding of the opportunities for, and benefits of, implementing large-scale data management systems using fully distributed, scalable and reliable components with standard interfaces. In particular, the discussion revealed that there are many ongoing activities in EC projects, and in the market, related to this, and there is definitely a willingness amongst the experts on data management systems to include object stores in their services.

In addition, our interactions with user communities confirmed that there is a large demand for future-proof, scalable, reliable and cost effective data management solutions at a scale that cannot be easily addressed by current infrastructures. For instance Europeana and the fusion research communities themselves have needs that result from the immense volume of data that they need to handle, as well as from the complexity of the structure of their data sets, including the relationships between the data objects, and also from the rates required for data access and storage – these are large enough to challenge traditional architectures based on file systems and transactional databases.

In parallel to these activities, we conducted an extended proof of concept of a B2SAFE-like data storage and access service. This work demonstrated that existing object stores implementations have the potential to provide a solid basis for the fundamental EUDAT CDI functionality related to data storage, access and management (such as ensuring the replication of data and the integrity of the copies, as well as PID assignment).

This part of our activities focused on the immediate applicability of the concepts, and on the architectures proposed and considered by us to be relevant to the current and short-term needs of the EUDAT CDI.

From a broader perspective, our work lead us to the belief that the future CDI has to be re-designed towards full distribution and decoupling its functional components. This will enable real breakthroughs in scalability, and future-proofing, and will make it possible to address the expected future growth of data volumes and I/O traffic.

Technology-wise, greater distribution of the EUDAT CDI services is already possible today thanks to the availability of relevant components, such as object stores, that can be applied for scalable data persistency and access. However, the re-design process should be conducted holistically, including all the system levels: storage and access, data persistency, data repositories and user facing applications, as well as the EUDAT data model.

For instance, achieving the full distribution of data management needed to further scale the I/O performance might require adopting data consistency models that are more relaxed than the ones that have been used for years for reliable data management. Similarly, applying graph technologies to represent object relationships might be needed to improve data search and exploration possibilities, compared to the relational databases typically used in metadata management.

Such fundamental changes in the system design and data model would impact the overall structure of the CDI and the implementation of particular services. Therefore, as these services have to stay productive and already hold data for lots of users, we do not expect this process to happen within the EUDAT2020 project.

5.2. Recommendations for Future Architecture of EUDAT

In this section we summarize the main assumptions of the envisioned future architecture of EUDAT. As the overall design of the CDI requires the involvement of all the stakeholders in EUDAT, our proposal covers target features of the architecture of the CDI and its components, rather than providing a final and complete view.

First of all, according to results of our work, the architecture should be decentralized. While this can be achieved only by conducting a comprehensive re-design process spanning all aspects of the EUDAT CDI, at least two basic aspects should be considered, from our point of view.

On one hand, particular layers should employ fully distributed implementations. Within the scope of our work we have concluded that applying object stores or other similarly distributed storage systems as the basic data storage and persistency layer would be profitable. In addition, based on the work in T9.1.2 related to Dynafed, we predict that enabling a relevant, dynamic data access federation solution would improve scalability, transparency and reliability of the data access vs what is currently possible based on static assignment of the storage sites to client communities.

On the other hand, the overall architecture of the CDI, as well as the organization of the data management, storage and access processes should be decentralized and decoupled from the conceptual and technical points of view.

For instance, data handling policies should be defined in the DPM at a high, technology-agnostic level. Steps in this direction have been made already with the current DPM implementation. However, policies are still relatively bound to iRODS-derived concepts, as well as to physical features of the EUDAT installations, such as the IP addresses of the endpoints. Therefore the DPM requires further work in order to make it possible to abstract user-level policies from the internal technological and organizational details of EUDAT. It should be possible to implement actual data management processes based on various technologies, including legacy systems (for example, filesystems applied for data, or relational database management systems for metadata) and modern solutions (such as object stores for data, graph data bases for meta-data, and HTTP federations for access).

In several use cases associated with EUDAT, it is acceptable to perform data management actions asynchronously to the data storage and access. In these cases, decentralization and decoupling of data management processes could bring additional benefits as replication could be performed by other system components, while storage and access can be effectively served by the user front-end elements.

Asynchronous implementation of users' I/O and data replication processes can be achieved if the data consistency guarantees in the system are relaxed in comparison to in the transactional model. In the use-cases typical to the EUDAT research communities and those aiming to exploit the CDI, the eventual consistency model is suitable, as the system is supposed to handle mainly archival and immutable content (with the exception of B2DROP). As already pointed out, adopting this model makes achieving scalability of system capacity and performance possible at relatively low costs.

Second, higher modularity of the EUDAT CDI architecture and infrastructure should be achieved. Functional components have to be further decomposed and encapsulated into fine-grained modules with standard

interfaces such as S3. In the long term, the concept of micro-services-based architecture should be considered. The actual degree of decomposition and modularizing of the EUDAT CDI architecture should be a matter of further discussions across the consortium, however it is already clear that several types of functionality should be encompassed in the common, shared system layers.

The data storage, access and persistency layer is one of the basic, self-contained and architecturally independent CDI components for which a universal, standard compliant interface could be defined. It can be integrated through plug-ins with external functionality, such as PID management. It could also be exploited as the common storage back-end for high-level data storage and access, data preservation, sharing, publication and synchronization services such as B2SAFE, B2SHARE, B2DROP or others. This layer should implement basic data management processes, such as ensuring data availability, data storage redundancy, data integrity and maintaining and handling data and metadata relationships. These processes could be steered, based on relatively abstract, technology-agnostic policies defined in the DPM.

The overall architecture of the envisioned future CDI is presented in Figure 3 below. The level of generality of the design assumptions described previously, as well as the draft scheme presented below, is intentionally high, as the details would require broad discussion and consensus related to the target CDI features and intended future use-cases as well as directions of its envisioned evolution. However, in principle, the architecture should ensure flexibility while adding user-facing services on the one hand, and unification and integration of the methods and protocols used for interacting with infrastructure back-ends on the other. It is possible to implement diverse functionalities based on the common, unified back-end, which should span services for long-term storage and preservation (B2SAFE) and publication (B2SHARE), as well as solutions and tools for short-term storage, the sharing and exchange of non-registered data (B2DROP), while exploiting a common, scalable data storage and access back-end.

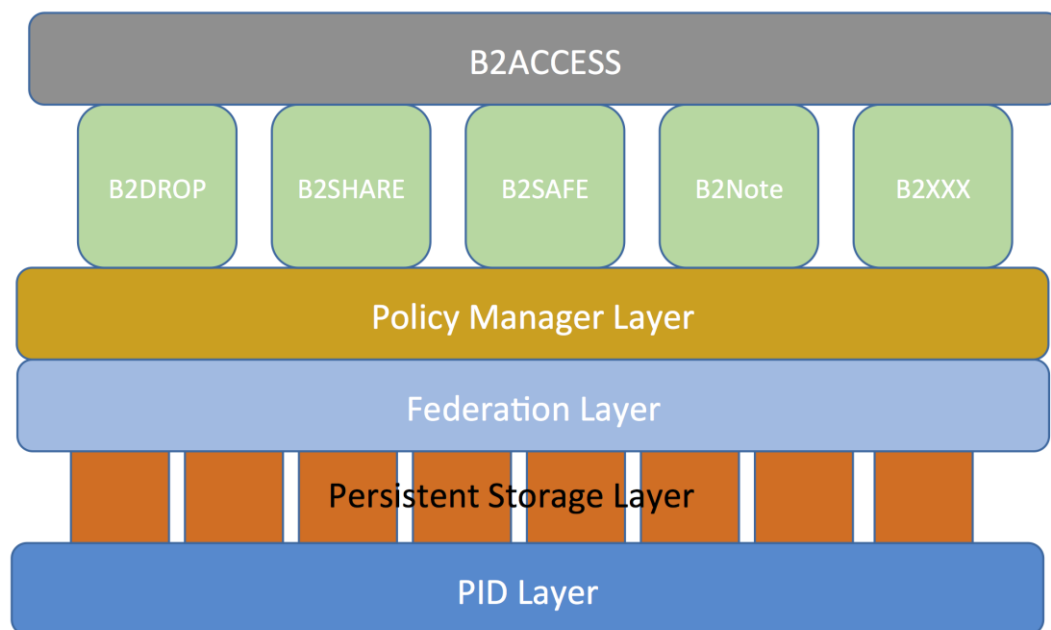


Figure 3: Envisioned future architecture of EUDAT CDI

It is important to note that, realistically, different kinds of low-level data storage, access and management systems will co-exist in the EUDAT infrastructure for several years. Therefore, an approach including these various systems as back-ends for a common set of high-level services should also be worked out.

Integrating them behind a centralized federation of components such as iRODS is not a credible solution for future according to our research and the feedback from our consultations. Instead, unifying and federating them on the protocol and API-level should be performed.

Although work on enabling REST APIs for iRODS-based B2SAFE was conducted, nevertheless such an approach only provides compatibility of iRODS with services that require REST API to integrate with storage back-end. It also conserves the centralization of the I/O path towards storage resources behind iRODS.

Therefore it is proposed to provide a cloud storage-based service in addition to the iRODS-based B2SAFE. In particular the S3-speaking B2SAFE-light service may be offered based on object storage systems, graph databases and HTTP-based access federations, in parallel to the B2SAFE implementation based on iRODS, relational databases and file systems. Such an approach will enable exploiting the full potential of modern object storage based technologies, while keeping possibility to use iRODS-based version wherever needed.

The object storage based B2SAFE-light would provide high-performance with regard to storage and access, including the capability to handle millions of digital objects that might be unaffordable in terms of classical storage and metadata handling solutions. In the same time, iRODS-based B2SAFE would be able to ensure transactional integrity of the data and metadata and provide ease of integration with existing, POSIX-based storage systems at data centres.

5.3. Summary

In the previous section we included a set of recommendations related to the envisioned future architecture of the EUDAT CDI and presented several possible ways to cope with the diversity of today's, and the envisioned, implementations of the basic storage and data-management layers.

Importantly, neither our task T9.1.1, nor the WP9 JRA in general, is in a position to take strategic decisions related to adopting any element of the proposed concepts or approaches for the further development of the CDI. Instead we are tasked with informing the EUDAT Technical Committee regarding advances and prototyping functionality. Therefore we consider our mission complete and concluded in this deliverable.

We are aware that adoption of object stores into the EUDAT portfolio is dependent on the level of the deployment of object stores at partnering sites. However, we also see that object storage systems are becoming widespread across the academic data centres, and therefore we believe the results of our work will be useful in the near future, both in the context of the EUDAT CDI and beyond.

5.4. Possible Further Work

The work reported in this document fills the obligations defined in the DoA of the EUDAT2020 project. In this section we overview possible future work on exploring object storage-related concepts and potential directions of collaboration within WP9 and other activities in EUDAT, aimed at providing comprehensive and integrated solution for data storage, access and management.

Demonstration of the full potential of modern, fully distributed, HTTP-based technologies in the area of high-performance and reliable, long-term data storage and management as well as federated, transparent data access would require developing and integrating several components.

First, enabling cross-technology data management, in particular data replication, would be an interesting extension of the work conducted in T9.1.1. So far we have used the built-in functionality of object stores in order to implement data replication among multiple OpenStack Swift clusters. While they were operated in distinct administrative domains, technology-wise they were compatible. These clusters were replicating data to each other using built-in OpenStack Swift container synchronisation mechanism. The extension of the current implementation would be to provide data management mechanisms that works cross-technology, possibly based on external replication mechanisms, triggered by the object storage system that receives the data sets from the clients and serves other I/O requests that modify the content of the object store. Taking into account that future EUDAT CDI might include sites operating object stores based on various technologies, implementing data management processes so that they can work cross-technology would be useful.

Second, support for the S3 protocol on the front-end of the DynaFed would enable federating access to existing S3-enabled object stores transparently from the point of view of the access protocol. In particular,

such solution will enable users to keep using S3 protocol in order to interact directly with actual storage endpoint while in the same time offering an option to interact with multiple storage endpoints through a S3-speaking federation. This should facilitate adoption of the federated object storage for the existing applications as well as within software stacks of the EUDAT CDI services as users and application developers would not be forced to use another, custom protocol in order to gain benefits brought by access federation. Such an activity is agreed to be an aim of the work within T9.1.2.

Third, the concept of integrated object stores and HTTP-based access federation (discussed in section 3.1.2) could be materialized based on the results of above mentioned extensions of the current work. In result, object stores would provide the basic data storage and access functionality as well as data management, including cross-technology replication and PID creation. In turn, HTTP-based, S3-enabled DynaFed federation would offer the transparent access to distributed data replicas, held in geographically distinct and technologically different object storage systems, through a S3 protocol that is being adopted by users and service developers as a common way of accessing scalable storage systems.