SEVENTH FRAMEWORK PROGRAMME

ForgetIT

www.forgetit-project.eu

ForgetIT

Concise Preservation by Combining Managed Forgetting and Contextualization Remembering

Grant Agreement No. 600826

Deliverable D7.1

Work-package	WP7: Computational Storage Services
Deliverable	D7.1: Foundations of Computational Storage Services
Deliverable Leader	Ealan Henis, IBM
Quality Assessor	Olivier Dobberkau olivier.dobberkau@dkd.de
Estimation of PM spent	4
Dissemination level	PU
Delivery date in Annex I	M6 (July 2013)
Actual delivery date	July 31, 2013
Revisions	7
Status	Final
Keywords:	Computational Storage, Storlet, Preservation DataStores, Cloud Storage, Preservation aware storage, Long Term Digital Preservation

Disclaimer

List of Authors

Partner Acronym	Authors
IBM	**Ealan Henis, Simona Cohen**

Table of Contents

Figures

Executive summary

Today's cloud storage systems are designed for general use and not particularly for long term digital preservation. To maintain understanding of the content of the data over time automatic (and periodic) processes need to be applied, preferably at the storage level (storlets). The objectives of work package 7 are to analyze how storage level processing can be defined and executed within cloud object storage for preservation systems, and to extend current cloud technology to support computational storage.

This deliverable (D7.1) presents a brief description of the foundations of computational storage services, in the context of the ForgetIT goals. It includes a survey on the state of the art in relevant fields, gap analysis and initial architecture and directions for the storlet-enabled preservation store. This document serves as the starting point for defining the architecture, extensions and interface for ForgetIT computational storage services implementation, to be provided later in deliverables D7.2, D7.3, D7.4).

We begin with a summary of the ForgetIT project goals and the specific goals addressed by Work Package 7 "Computational Storage Services". It aims to provide computations in the storage (storlets) to support Preservation Aware Storage and the goal of maintaining data readable and useful over long periods of time. The general approach taken in Work Package 7 is to leverage previous work (Preservation DataStores – abbreviated PDS - and Preservation DataStores in the Cloud) and extend/adapt these technologies for the needs of the ForgetIT project via building a computational storage mechanism termed "Storlet Engine". The Storlet Engine will allow performing ForgetIT relevant data intensive operations near the data, at the storage server itself. This enables offloading preservation functionality to the storage and supports automation of archiving processes. It decreases the probability of data loss, simplifies the applications, and provides improved performance and robustness. The functions of the planned preservation-aware storage will be demonstrated with ForgetIT use cases. The planned mechanism is based on commercial cloud storage (OpenStack Swift), enhanced with special dedicated data model and interfaces for storlets definition, deployment and execution.

In order to provide the background to our approach and the required extensions, we provide below a brief review of existing relevant technologies, and the ForgetIT preservation storage system solution approach. We start by describing the basics of Long Term Digital Preservation (LTDP), followed by relevant storage cloud technologies and gap analysis between existing and desired functions. We provide a more detailed description of PDS, PDS Cloud and the existing Storlet Engine, since we plan to use these as our technology starting point. Next, and as the next logical step, we point to initial design directions for the ForgetIT Workpackage 7, addressing the stated Work Package goals.

We provide details on the planned Storlet Engine extensions for ForgetIT, and list examples of potential storlets that are expected to support the ForgetIT goals. We outline the planned integration of our technology and mechanism with ForgetIT modules of other Work Packages.

Finally, we mention additional possible directions that we may pursue in addition to the main goals described.

1 Introduction

1.1 ForgetIT Goals

Due to the increase in digital content creation on one hand, and the lack of systematic preservation mechanisms on the other, a significant danger is introduced for losing data in the long run. While Long Term Digital Preservation (LTDP) of digital content is already well established in institutions such as national libraries and archives, it is yet to be adopted by other organizations, and even more so for personal content.

The goal of the ForgetIT project is to ease the adoption of preservation in the personal and organizational context by building a data preservation system that includes a managed forgetting mechanism. Since forgetting plays a crucial role for humans, it is argued that managed forgetting of saved digital data is a viable alternative to keeping an ever growing amount of data indefinitely. Targeting personal and organizational use cases, the approach used is to apply contextualized remembering, synergetic preservation and managed forgetting. The planned methodology will combine LTDP, information management, multimedia analysis, personal information management, storage computing, cloud computing, cognitive psychology, law, and economics. The selected ForgetIT means to achieve these goals combine three new concepts: 1) Resource selection via managed forgetting modeling, 2) Synergetic Preservation as an integral part of the information content lifecycle management and 3) Contextualized Remembering that is evolution-aware.

An important layer in the ForgetIT architecture is a preservation aware storage with a Storlet Engine for processing automatic computations close to the data.

1.2 ForgetIT Preservation Aware Computational Storage Goals

While preserving organizational, public and personal assets for a long term involves various technologies, storage has a key role since it stores the data for most of its lifecycle. Despite the increase in the ability to store digital data, the ability to maintain these data readable and useful over time decreases (for example, due to frequent changes in rendering technologies). Consequently, the goals of work package 7 aimed to support the ForgetIT general goals by building a preservation aware computational storage system for the ForgetIT framework (see red circle in ForgetIT framework figure below). The approach is to study, analyze and implement a computational storage mechanism (Storlet Engine, see more details below) for an LTDP storage system. This enables offloading preservation functionality to the storage and supports automation of archiving processes.

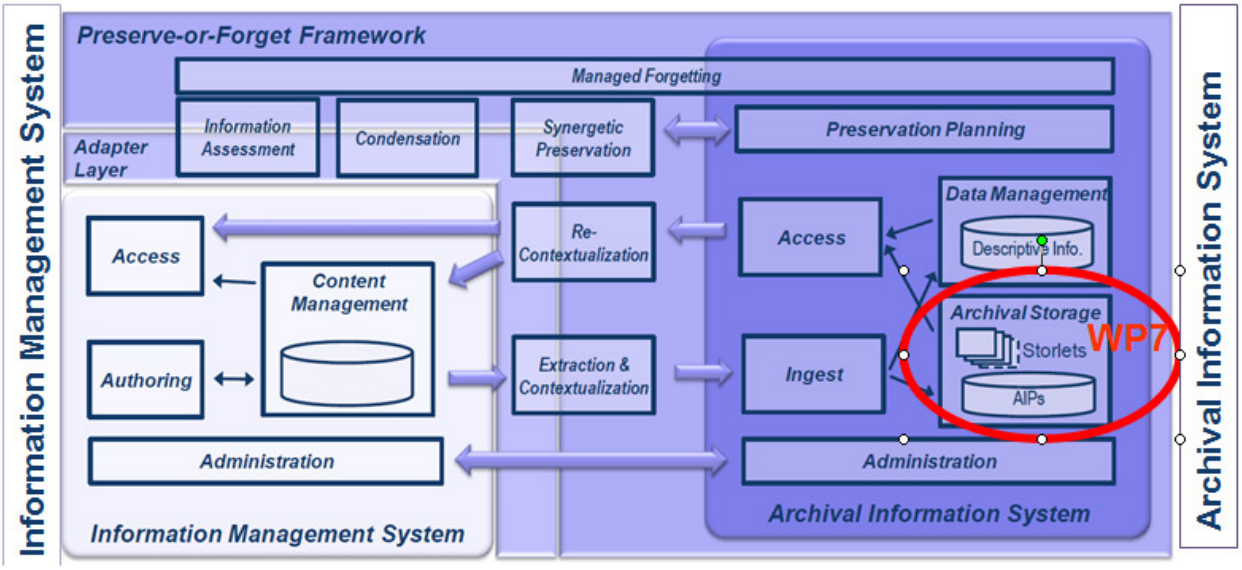

Figure 1: Preservation aware computational storage in ForgetIT framework

The functions of the planned preservation-aware computational storage will be demonstrated with ForgetIT use cases (relevant data and computations). Hence, we plan to build a consolidated platform for storage objects and computational processes (storlets) that may be defined, triggered and executed close to the data. This will add flexibility to the storage service and allow extending its capabilities over time.

In the framework of work package 7, we plan to design a consolidated object and compute architecture that is based on commercial cloud storage, and enhance it with a dedicated data model and interfaces for storlets. These will be focused on storlets that are relevant to the ForgetIT use cases (e.g., in cases which are data intensive or highly secure). We wish to leverage the outcomes of previous FP7 ENSURE and VISION Cloud projects to support the ForgetIT use cases. We also wish to extend the object interface to enable dynamically adding and removing storlets.

Since the storlets execution environment has diverse requirements (especially when used over time in various preservation environments), we also wish to investigate advanced challenges including the limitations of the storlet sandbox and their enforcement, the appropriate security model, handling of streaming data cases, storlet placement optimization and parallel and distributed execution.

1.3 Summary of existing relevant technologies

For completeness, a brief summary of relevant technologies is presented here, more details on Preservation DataStores (PDS) and storage clouds are given below in Sections 2 and 3.

Long Term Digital Preservation (LTDP) is the ability to sustain the understandability and usability of digital objects in the future regardless of changes in technologies and designated communities that create and consume these digital objects. The core standard in LTDP is Open Archival Information System (OAIS), an ISO standard since 2003, revised in 2012 (ISO 14721:2012) [1]. OAIS specifies the terms, concepts and reference models for a system dedicated to preserving digital assets for a designated community. OAIS defines a functional model, entities and the flows among them. The basic stored object is Archival Information Package (AIP). It is a composite object that includes raw data and additional metadata needed to interpret the data in the (long term) future.

Preservation DataStores (PDS) is an OAIS-based preservation aware storage system. It supports both bit preservation (the ability to retrieve the bits in the face of physical media degradation or obsolescence, corruption or destruction) as well as logical preservation (preserving the understandability and usability of the data, despite changes that will take place in servers, operating systems, data management, products, applications).

PDS packages raw data with its metadata into a self-contained AIP, logs its provenance, and assures its integrity (typically by calculating its signature – fixity). It handles AIP repackaging and transformation. One of the innovative features of PDS is that it offloads computation to the underlying storage to decrease probability of data loss, simplify the applications, and provide improved performance and robustness. It uses a Storlet Engine that manages, deploys and executes storlets - restricted computation modules that run on storage systems close to the data that it processes. PDS uses the Storlet Engine to perform periodic or on-demand integrity checks, format transformations, analytics and other operations close to the data.

Storage Clouds (public or private) are recent models for storage provisioning. They are highly distributed (employ multiple storage nodes), scalable and virtualized environments that offer availability, accessibility and sharing of data at low cost. They focus on storage and data services, typically being agnostic to the content of the data they service. They emphasize availability to the data, some reliability and scalability, all at competitive costs.

One example of cloud storage system is the OpenStack Swift object storage [6]. OpenStack is an emerging open source software for building cloud computing platform for public and private clouds, with a huge amount of interest and rapidly growing community. The OpenStack software includes the Swift object storage that we will utilize in work package 7.

1.4 The ForgetIT Preservation Storage System Solution Approach

Storlets are a good candidate for providing automated reliable periodic or on-demand computational processes executed at low cost. To meet the ForgetIT goals, we wish to exploit the storage node processing capabilities and extend the PDS storlets (computation modules that run on the storage servers close to the data) to handle typical ForgetIT use cases.

Server-based object storage systems are ideal for executing storlets. They typically need to serve large data sets accessed from anywhere over the WAN. Also, the commodity servers typically have powerful CPUs that are underutilized. Exploiting the unused CPU cycles for executing storlets within the data residency is beneficial, and the savings of bandwidth and infrastructure are expected to be significant. Additionally, security and compliance are improved. The best candidate cloud storage for our proposed LTPD storage framework is OpenStack Swift (see more details below). However, current cloud storage platforms are designed for general use and are not particularly suited for LTDP. For example, automated reliable versatile periodic integrity checks at the data level are missing. Hence, we opted to enhance existing open cloud storage platforms (e.g., OpenStack) with various new capabilities, to enable new storage level automated computational processes as inspired by the ForgetIT use cases. Some examples of new planned processes are: format transformations (for long-term availability), summarization (as part of managed forgetting), redundancy detection etc.

In order to define the required enhancements to existing cloud storage, we summarize a gap analysis between the capabilities of current systems and those required by the ForgetIT preservation storage system.

2 Survey of State of the Art of Long Term Digital Preservation

While the state of the art in long term digital preservation is extensive and includes several technologies and EU projects, this chapter is mainly focused on the state of the art that relates to storage.

2.1 Long Term Digital Preservation

The growing need to preserve large volumes of digital content for decades by organizations and private citizens is further increased by regulatory compliance and legal issues. Data types that need to be preserved include mail, medical records, financial accounts, aircraft designs, etc.

Long Term Digital Preservation (LTDP) is defined as the ability to sustain the understandability and usability of digital objects in the distant future regardless of changes in technologies and in the designated communities that create and/or consume these digital objects. LTDP is a special case of digital archiving, where the lifetime of the stored data may exceed the lifetime of the application and format used to create and interpret the data. Consequently, in addition to its legacy roles, preservation-aware storage must handle also repackaging and data transformation.

LTDP includes bit preservation and logical preservation. Bit preservation is the ability to retrieve the bits in the face of physical media degradation or obsolescence, corruption or destruction due to errors or malicious attacks, or even environmental catastrophes such as fire and flooding. Logical preservation means preserving the understandability and usability of the data, despite changes in servers, operating systems, data management products, data formats applications and even users over the long term. Additionally, logical preservation requires maintaining the provenance of the data, along with its authenticity and integrity, so it may be verified that only legitimate users have accessed the data and that the data is reliable and trustworthy.

Logical preservation is a challenging open research area attempting to enable future understanding of the preserved data by future technologies and by future users that may hold a different knowledge base from that of the data producers. To support logical preservation, preservation-aware storage needs to associate the raw data with metadata that describes its context, logs its provenance, and assures its fixity, perform data transformation, replace obsolete and add new formats. Hence, the LTDP storage subsystem should attempt to encapsulate the raw data with its complex interrelated metadata objects.

The core current standard for LTDP is Open Archival Information System (OAIS).

2.2 The core standard for LTDP: Open Archival Information System (OAIS)

OAIS is an ISO standard for LTDP since 2003, revised in 2012 (ISO 14721:2012) [1]. It specifies the terms, concepts and reference models for a system dedicated to preserving digital assets for a designated community. OAIS defines a functional model and the basic stored objects. The entities of the functional model and the flows among them are depicted in Figure 1.

Our technology in work package 7 for preservation aware computational storage includes PDS in the cloud with the Storlet Engine. This technology takes the role of the archival storage entity in the OAIS functional model.

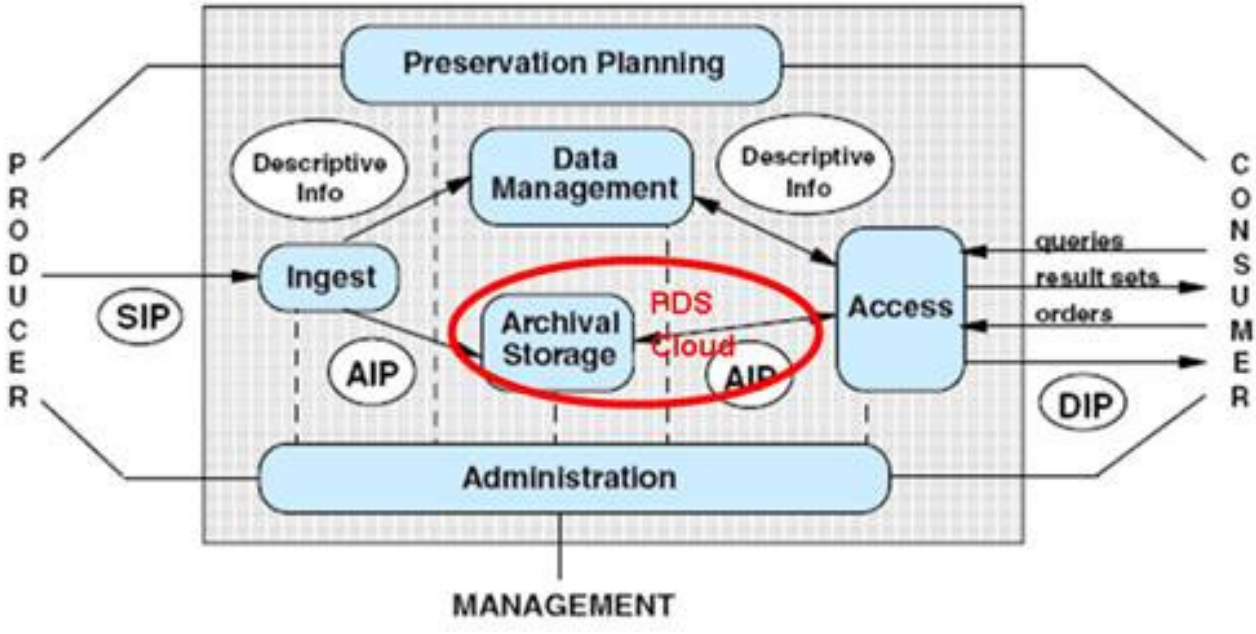

Figure 2: OAIS functional model and PDS

The basic high level structure for stored OAIS object is the Archival Information Package (AIP). It is a composite object that includes raw data and additional metadata. An AIP contains the content information, including the Content Data Object or raw data which is the focus of the preservation, as well as the recursive Representation Information (RepInfo) needed to render the object intelligible to its designated community. The representation information may include information about the hardware and software environment needed to view and interpret the content data object.

The other part of an AIP is the Preservation Description Information (PDI), which is further divided into four sections:

- Reference — globally unique and persistent identifiers for the content data object
- Provenance — chain of custody, the history and the origin of the content information custody
- Context — relationships of the content information to its environment
- Fixity — a demonstration that the particular content information has not been altered in an undocumented manner

The OAIS AIP logical structure is depicted in the figure below

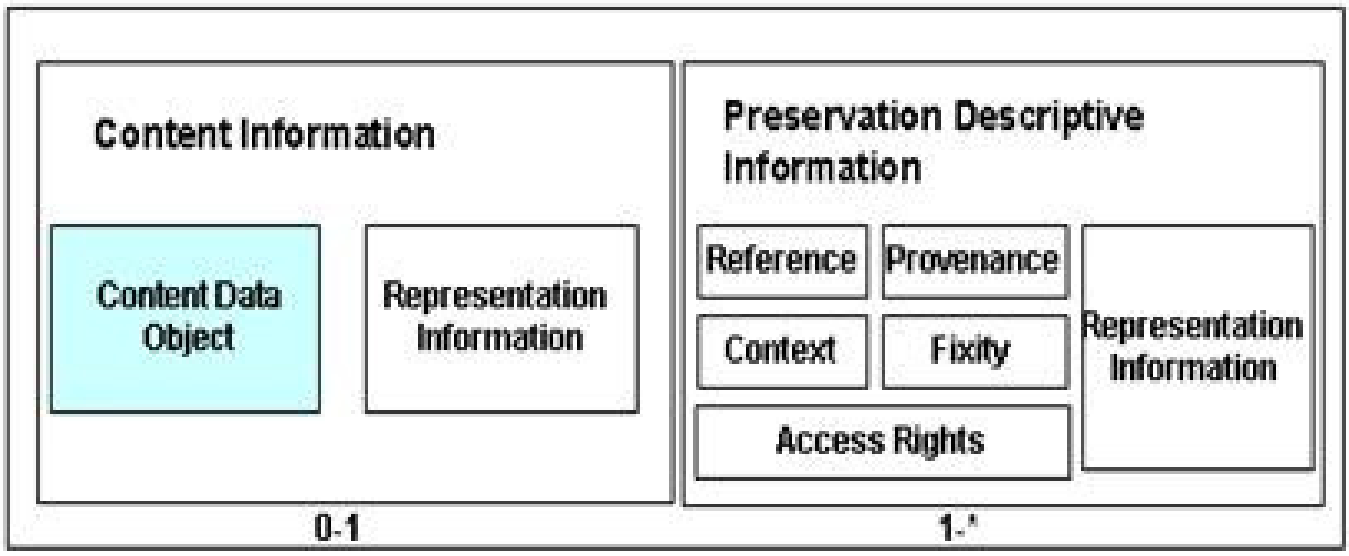

Figure 3: OAIS AIP logical structure

The AIP metadata are essential for logical preservation of the raw data, and one important metadata is generally termed Representation Information (RepInfo). The RepInfo themselves are data objects, hence, they are also stored as AIPs, and recursively have their own RepInfo etc., creating a RepInfo network. The recursion ends when an agreed common knowledge base of the designated community that uses the data is reached. As improved preservation aware storage subsystems emerge, it will become possible to incorporate some of the OAIS processes into them.

2.2.1 Archival Storage

The Archival Storage entity (see OAIS functional model figure) provides the services and functions for the storage, maintenance and retrieval of AIPs. Archival Storage functions include: receiving AIPs from Ingest requests and providing AIPs to Access requests, storing the AIPs, managing the storage hierarchy, refreshing the media, performing routine and special error checking, and providing disaster recovery capabilities.

The Receive Data function receives a storage request and an AIP from an Ingest request and stores the AIP on permanent storage within the archive. The Ingest request may indicate data object features like anticipated frequency of utilization of the data, in order to allow for appropriate data placement for the AIP (e.g., type of media).

The Provide Data function provides copies of stored AIPs in response to Access requests.

The Manage Storage Hierarchy function places the AIP on the appropriate media, based on storage management policies, operational statistics, or directions from Ingest request. These include on-line, off-line or near-line storage, required throughput rate, maximum allowed bit error rate, or special handling or backup procedures. This function also provides operational statistics.

The Replace Media function provides the capability to reproduce the AIPs over time on another media. The migration selects a storage medium, taking into consideration the expected and actual rates of errors encountered in various media types, their performance, and their costs of ownership. This function may perform Refreshment, Replication, and Repackaging (e.g., migration to new media under a new operating system and file system).

The Error Checking function provides statistically acceptable assurance that no components of the AIP are corrupted during any internal Archival Storage data transfer. A standard mechanism for tracking and verifying the validity of all data objects within the archive may also be used (e.g., CRCs or some other error checking mechanism).

The Disaster Recovery function provides a mechanism for duplicating the digital contents of the archive collection and storing the duplicate in a physically separate facility.

2.3 Self-contained Information Retention Format (SIRF)

The Self-contained Information Retention Format (SIRF) is a storage container format for preservation objects that provides a catalog with metadata related to the entire contents of the container, as well as to the individual objects and their interrelationship.

Recognizing the significance of standardization in storage for long term retention, the Storage Networking Industry Association (SNIA) made progress towards this goal. A survey with more than 250 participants showed that over 80% of respondents had retention requirements of at least 50 years [40]. The top external factors driving these retention requirements were mainly business risks and compliance with regulations. Consequently, SNIA formed the Long Term Retention (LTR) Technical Working Group (TWG) (http://www.snia.org/ltr) in 2008 to address storage aspects of digital retention. The mission of LTR, which is co-led by IBM Research-Haifa and HP, is to lead storage industry collaboration, and develop technologies, models, educational materials and practices related to retention and preservation.

The LTR TWG is working on Self-contained Information Retention Format (SIRF) to create a standardized vendor-neutral storage format that will allow interpreting preservation objects in the future, even by systems and applications that do not exist today. SIRF provides strong encapsulation of large quantities of metadata with the data at the storage level, and enables easy migration of the preserved data across storage devices. In contrast to the semantics of traditional file systems, which include only limited metadata for each file, SIRF provides rich metadata needed for preservation and ensures its grouping with the information objects.

Archivists and records managers of physical items such as documents, records, etc., avoid processing each item individually. Typically, they gather together a group of items that are related in some manner - by usage, by association with a specific event, by timing, etc. - and then perform all of the processing on the group. The group itself may be known as a series, a collection, or in some cases as a record or a record group. Once assembled, an archivist will typically place the series in a physical container (e.g., a file folder or a filing box of standard dimensions), stick a label that marks the container with a name and a reference number, and place the container in a known location.

SIRF [41][42] proposes an approach to digital content preservation that leverages the processes of the archival profession, thus helping archivists remain comfortable with the digital domain. One of the major needs that will make this strategy usable is the availability of a digital equivalent to the physical container. This archival box or file folder defines a set or series, and can be labeled with standard information in a defined format to allow retrieval when needed. SIRF is intended to be that equivalent - a storage container format for a set of (digital) preservation objects that also provides a catalog with metadata related to the entire contents of the container, as well as to the individual objects and their interrelationship. This logical container makes it easier and more efficient to provide many of the processes that are needed to address threats to the digital content. Easier and efficient preservation processes, in turn, lead to an increase in scalability and reduction in cost for preservation of digital content.

The following figure illustrates the components included in the SIRF container:

- A magic object that identifies whether this is a SIRF container and gives its version.
- Preservation objects that are immutable. The container may include multiple versions of a preservation object.
- A catalog that is updateable and contains metadata needed to make the container and its preservation objects portable into the future without relying on functions external to the storage subsystem. It contains metadata relating to the entire contents of the container as well as to the individual preservation objects and their relationships.

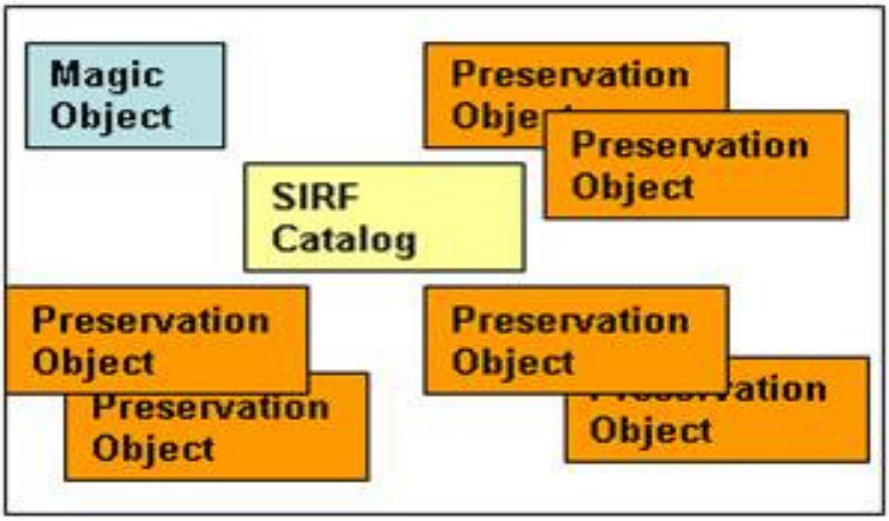

Figure 4: SIRF container

Both cloud and tape technologies are viable alternatives for storage of big data. SNIA supports their standardization. The SNIA Cloud Data Management Interface (CDMI) provides a standardized interface to create, retrieve, update, and delete objects in a cloud. The SNIA Linear Tape File System (LTFS) takes advantage of a new generation of tape hardware to provide efficient access to tape using standard, familiar system tools and interfaces. The LTR TWG also works on how to combine SIRF with LTFS and SIRF with CDMI to address LTDP challenges.

2.4 Previous Works Related to LTDP Storage

Baker et al. [24][25] suggest bit preservation guidelines and alternative architectural solutions that focus on replication across autonomous sites, reduced per-site engineering costs, and the ability to scale over time and technologies.

Dappert and Enders [27] discuss the importance of metadata in a long term preservation solution. The authors identify several categories of metadata, including descriptive, preservation related, and structural, arguing that no single existing metadata schema accommodates the representation of all categories. The work surveys metadata specifications contributing to long term preservation.

Rhea et al. [38] describe an archival storage layer which uses content-addressed storage (CAS) to retain nightly snapshots of users' disks indefinitely. In addition, they also snapshot the entire virtualized desktop. Their focus is on efficiently leveraging on-the-shelf storage to create a personalized archive. Arguably, preserving the entire virtualized desktop does not scale or age well.

With regard to the more general storage aspects of digital preservation, previous works [24][25][43] address authentication as well as security issues. Some works [24] explore the needs of long-term storage and present a reliability model and associated strategies and architectures.

The e-depot digital archiving system of the National Library of the Netherlands (KB) [44] is composed of the Digital Information Archiving System (DIAS). The e-depot library conforms to the OAIS standard, and addresses both bit and logical preservation.

Provenance-aware storage system (PASS) [45] and data modeling for provenance [46] track the provenance of data at the storage level. The provenance of data is tracked at the file system level, and does not employ an auxiliary database for provenance management. They suggest offloading storage related activities to the

storage level (similar to the PDS work [3][4][15][21][26] – see below). In [46] the authors argue that due to the common ancestry relations of provenance data, these data naturally form a directed graph. Hence, provenance data and query models should address this structure. A semi-structured data model with a special query language (PQL) was used, taken from the object oriented database community.

The LOCKSS [48] system (Lots of Copies Keep Stuff Safe) generates several copies of each object to enhance its resiliency to data loss.

The Storage Resource Broker (SRB) [49] developed by the San Diego Supercomputing Centre (SDSC) is a logical distributed file system with a client-server architecture. It presents the user with a single global logical namespace or file hierarchy. SRB is a data grid technology middleware built on top of file systems, archives, real-time data sources, and storage systems. Several preservation projects, such as the preservation environment for the National Archives and Records Administration (NARA), utilize the SRB as the foundation for their archives. The SRB code was recently extended into a new system called iRODS [50], Intelligent Rule-Oriented Data management System. The iRODS rule engine can be useful for implementing preservation-specific policies.

Venti [47] is a block-level network storage system intended for archival data. In itself, it does not provide the services of a file or backup system, rather the backend archival storage for these types of applications. In the system, a 160-bit SHA-1 hash of the data (called a fingerprint) acts as the address of the data, which enforces a write-once policy. Data blocks cannot be removed; making Venti specialized for permanent or backup storage. Venti provides inherent integrity checking of data by computing the fingerprint on the retrieved data and comparing it to the fingerprint held by the application.

You et al. [31] presented PRESIDIO: a scalable archival storage system that detects similarity and attempts to reduce used space.

Deep Store [51] is archival storage architecture for a large-scale storage system capable of efficiently and reliably storing immutable data. The Deep Store model for storage uses an architecture consisting of four primary abstractions: storage objects, physical storage components, software architecture, and a storage interface. Space efficiency is achieved by a content analysis phase, which includes PRESIDIO, a new lossless data compression framework that incorporates inter-file compression with a content-addressable object storage mechanism. The reliability of files with shared data is provided by using variable levels of redundancy. The metadata is stored in XML format and methods for searching and versioning the metadata are provided.

The OCLC Digital Archive [29] provides a secure storage environment for master files and digital originals for libraries (backups, disaster recovery, periodic fixity check, virus check, manifest and format verification).

Bessani et. al. [30] presented DepSky: a storage system in a cloud-of-clouds with a set of Byzantine quorum system protocols, cryptography, secret sharing, erasure codes. One of the key objectives of DepSky is to reduce cost. DepSky and PRESIDIO suggest advanced protocols and implementations for specific data (static) management functions.

Muniswamy-Reddy et al. [28] introduce protocols for cloud that collect and track the provenance of data objects.

Offloading data maintenance functions from the application to the storage system is an ongoing trend. Functionality such as bit-to-bit data migration, block level data integrity, and encryption are commonly carried out by advanced intelligent storage subsystems.

The requirements for preservation-aware storage are characterized in a position paper [26] and Preservation DataStores (PDS) [4][15][21] was such a system developed by IBM as part of the CASPAR EU project. It was later extended and adapted to a cloud version in PDS-Cloud [3] as part of the ENSURE EU project. PDS and PDS Cloud are OAIS based LTDP systems that provide logical preservation coupled with extended offloading of data related functions to the storage. Since these works form the basis for the ForgetIT

preservation-aware storage system, we devote separate Sections (Sections 3 and 4, see below) to a detailed description of PDS and PDS-Cloud.

A detailed description of the PDS provenance and authenticity management was presented in [52].

An extensive bibliography of digital preservation and curation has been compiled by Bailey [39]. It covers multiple subjects, including models, formats, ongoing projects and research.

2.5 Cloud Technologies and Gap Analysis

2.5.1 Cloud Technologies

2.5.1.1 General

Cloud technology is emerging as an infrastructure suitable for building large and complex systems. Storage and compute resources provisioned from converged infrastructure and shared resource pools present a cost effective alternative to the traditional in-house data center. The cloud provides new levels of scalability, elasticity and availability, and enables simple access to data from any location and any device.

LTDP can benefit from cloud technology. With its many vendors, open interfaces and subscription payment model, cloud storage offers the flexibility needed to address the dynamically evolving requirements of preservation. The potentially unlimited capacity provides inherent scalability and redundancy. In a preservation system, the preservation object (e.g., AIP) holds references (internal and to other objects on possibly other clouds). In cloud storage, objects exist within containers, which may be useful for referencing. The cloud typically provides a data model of objects that include raw data and user-defined key-value metadata pairs, treated as a unit.

Cloud storage is a good solution for latency-tolerant applications (e.g., backup and archiving), and for digital preservation repositories. Hence, the cloud is an attractive platform for building LTDP (e.g., OAIS based) storage and compute resources, provisioned from converged infrastructure and shared resource pools.

2.5.1.2 Cloud Data Management Interface (CDMI)

Efforts to unify cloud APIs across the multitude of vendors and to cater to diverse application requirements have lead to the specification of the CDMI (Cloud Data Management Interface) [19]. CDMI is a SNIA architecture standard.

2.5.1.3 Current Cloud Platforms

The basic features of all current Cloud platforms are similar. Some of the well known platforms are Amazon S3 [53] and EC2 (enterprise) [54], Openstack [6] Swift [55] and Nova (private, open source) [56], Windows Azure Storage [57], Eucalyptus [10], Rackspace [11], EMC Atmos [12], DuraCloud [13] open source, and the recent VISION Cloud [58]. All cloud platforms offer a RESTful [14] interface, and the URI path format is similar (although some HTTP headers are different). The security and authorization models vary among the platforms.

Amazon S3 has a two-level data organization: buckets and objects, up to 5 terabytes in size. Each object has system/user metadata name-value pairs.

In Windows Azure Storage data is in the form of Blobs within containers, two types of blobs are available: block blobs and page blobs (the latter are used as virtual hard drives). Page blobs maximum size is 1TB. Block and page blobs can have metadata name-value pairs.

Rackspace Cloud Files is powered by OpenStack3, uses REST APIs, and has a two level data model of objects and containers. Object size limit is 5GB, optional concatenation of objects via a manifest object is supported. The metadata for objects and containers is limited to 90 key-value pairs and total size of less than 4KB.

EMC's Atmos is a cloud storage platform to manage content with large amounts of data. It is not a clustered file system or a network attached storage product, but cloud-optimized storage. Data is distributed and accessed on a global scale. Scriptable policies are available for automated data placement and control of data handling. Metadata-driven policy-based management supports data placement, replication, protection, compression, de-duplication, retention (but not encryption). Replication may be synchronous or asynchronous. Atmos runs in a virtualized environment or within a specialized EMC sold hardware. Atmos has two major categories of metadata: system metadata generated automatically and cannot be modified by users, and user-defined metadata. Metadata may be non-listable or listable by queries or search. Search may only be performed on the metadata keys, not on the values. Key-value pairs are restricted to 1 KB in size. The user metadata can be used to define triggers for policies. Atmos uses REST and SOAP based APIs, and supports CIFS/NFS traditional file access. It has two data models: an object interface and a namespace interface, which is similar to a file system with folders. Atmos has ACLs to control data and metadata access.

The VISION Cloud approach extends EMC's Atmos with rich metadata, and supports active indexing inside the storage infrastructure. It extends metadata to support relations between objects, enabling construction of content networks dynamically and automatically by discovery. VISION uses storlets co-located with the data that can be activated by events such as uploading of a new object, adding metadata to it etc. VISION provides scalable, metadata-rich object services with a new data model not compatible with POSIX. It leverages distributed file systems rather than expand their capabilities. VISION addresses Big Data, managing (unstructured) metadata, content centric access – it provides APIs for retrieval and finding. VISION targets efficient management of metadata and scale, focuses on key-value stores and Big Table databases. VISION supports metadata search (both on keys and/or values) access, retrieval and statistics.

Metadata queries are not supported in Amazon S3, Google Cloud Storage, Windows Azure Storage, Rackspace Cloud Files. EMC's Atmos supports only queries on metadata keys not metadata values, queries on both metadata keys and values are supported by VISION.

DuraCloud's goal is a fully integrated environment of services and data across multiple cloud providers, supporting data storage, replication and access, as well as data preservation services such as format transformation and fixity checks. However, DuraCloud is not compliant with the OAIS reference model.

Swift and S3 do not support filtering (searching) of object by tags (metadata). The VISION Cloud, developed in the framework of the VISION EU project, provides some metadata query support.

An important difference between cloud platforms is their locality features, which affects their security: Swift is a private cloud and storage can be entirely under the control of a single client organization. On the other hand, S3 is a public service shared by many customers, and storage is located in the public domain.

Offloading some preservation maintenance procedures to the cloud requires compute cloud services along with the storage cloud. This compute/storage cloud option is supported by, e.g., Amazon EC2 and Openstack Nova.

2.5.2 Storage Clouds Gap Analysis for Supporting LTDP

With regard to usability for LTDP, many of the relevant features of the various cloud storage platforms are very similar. Their properties and deficiencies are given below.

2.5.2.1 Bit reliability: missing recurring fixity check and versatile algorithms

Guarantees of bit reliability in the cloud are insufficient for preservation systems. Storage cloud platforms generally perform a fixity check upon storing an object, but do not have an option to repeat this check periodically. Also, regulatory requirements for digital preservation may require performing fixity checks using multiple algorithms, whereas cloud platforms usually support one predefined method.

2.5.2.2 Data lock-in: cannot extract entire data

Cloud systems currently suffer from data lock-in, where there are no easy means to get the data out of the system in its entirety, reliably and efficiently. This poses a great risk as services providers may go out of business or become unreliable over time.

2.5.2.3 Certification and trust: no support for auditing certification and security compliance

Storage clouds lack support for auditing, certification and trust, including secure access. This is critical in preservation of commercial and business oriented data, in which evidence of regulatory compliance is required. Specifically, preservation related regulatory requirements entail support for data encryption, anonymization, periodic auditing, replication, versioning, etc.

2.5.2.4 Metadata: limited support in size, missing metadata search

One of the key concepts in the OAIS model for preservation is the extensive use of metadata, strongly coupled with the raw data as part of the AIP. Moreover, metadata is likely to change and grow significantly in size during the extended lifetime of the AIP. Storage clouds today have rather limited support of metadata. The allowable space for metadata (per object) is much too small for the extensive size of preservation metadata. Most clouds do not provide means to search their metadata, i.e., the ability to filter objects by tags.

2.5.2.5 Event tracking: partial only provenance

Storage clouds do not capture events that are part of the object provenance and need to be recorded for preservation, such as access to objects, media refresh events, etc. This is particularly crucial in the cloud, because data in the cloud can be widely shared. Provenance is critical for consumers to verify data authenticity and identity. It is of importance to keep the provenance together with the data, to guarantee consistency.

2.5.2.6 Workload management: need different cost model for LTDP

Currently, storage clouds are mostly used for backup or for applications requiring a high level of sharing, downloads and streaming. Therefore, the cost models today are well suited to this type of usage. Preservation systems pose a different type of workload, requirements, and SLAs resulting in possibly different cost models. For example, consumer access to preserved data may be infrequent, resulting in less demand for streaming and downloading. On the other hand, preservation maintenance may utilize more efficient (and less expensive) access to the preserved objects directly on the cloud, and this is typically performed in bulk on many objects.

2.5.2.7 Storage and compute synergy: lacking

Computational support is needed for preservation, as storage actions are performed over time. Data management functionalities may be offered transparently in the cloud (e.g., handling data replication and disaster recovery). The current cloud storage/compute synergy is insufficient for a digital preservation solution. Migration and transformation need to be more flexible and configurable as part of the Preservation Digital Asset Lifecycle (PDAL). Extended computations operated by the user in the storage/compute cloud are needed.

2.5.2.8 Logical preservation

As emphasized above, preservation is more than just ensuring the bit integrity of the object content. It must also support logical data preservation, such that the content is understandable in the future. In order to use cloud based preservation, a candidate cloud platform should be extended to bridge the above mentioned gaps, as current cloud environments do not provide built-in support for logical preservation.

As part of PDS Cloud [3], we have taken steps towards this goal. We have leveraged and extended the availability of storage/compute cloud synergy to perform cloud based data-related computations (Virtual Appliances – see details in Section 4 below) that may assist in the future interpretation and visualization of digital content, in support of logical preservation.

2.6 OpenStack Object Storage

OpenStack Object Storage (code-named Swift) is an open source software for creating redundant, fault-tolerant, eventually consistent object storage. It a distributed scalable system and uses clusters of standardized servers to store petabytes of accessible data. It is not a file system or real-time data storage system, but rather a long-term storage system for a more permanent type of static data that can be retrieved, leveraged, and then updated if necessary. Primary examples of data that best fit this type of storage model are virtual machine images, photo storage, email storage and backup archiving. Having no central “brain” or master point of control provides greater scalability, redundancy and permanence. Objects are written to multiple hardware devices in the data center, with the OpenStack software responsible for data replication and integrity across the cluster. Storage clusters can scale horizontally by adding new nodes. Should a node fail, OpenStack works to replicate its content from other active nodes.

The data model includes a hierarchy of accounts (tenants), containers and objects. The external interface to this data model is via the proxy server.

The Proxy Server is responsible for the external interface. It includes an HTTP server that implements the Swift RESTful API. It coordinates the read and write requests from clients and implements the read and write guarantees of the system. When a client sends a write request, the proxy ensures that the object has been successfully written to disk on the storage nodes (two of three replicas) before responding with a code indicating success. For each request, it will look up the location of the account, container, or object in the ring (see below) and route the request accordingly. It will also handle a large number of failures.
When objects are streamed to or from an object server, they are streamed directly through the proxy server to or from the user namely the proxy server does not spool them. The proxy services are more CPU and network I/O intensive. Several of the services rely on Memcached for caching certain types of lookups, such as auth tokens, and container/account existence.

The Object Server that exposes an internal RESTful API is a very simple blob storage server that can store, retrieve and delete objects stored on local devices. Each object is stored as a single binary file on disk, and object metadata is stored in the files extended attributes (xattrs). This simple design allows the objects data and metadata to be stored together and replicated as a single unit. Each object is stored using a path derived from the object names hash and the operations timestamp. The object services are more disk and network I/O intensive.

The Container Server exposes an internal RESTful API and its primary job is to handle listings of objects. It does not know where those objects are, just what objects are in a specific container. Although containers cannot be nested, they are conceptually similar to directories or folders in a file system. Users may set metadata on individual containers, and containers provide a listing of each object they contain. The listings are stored as sqlite database files, and replicated across the cluster similar to how objects are. There is no limit to the number of containers that a user may create within a swift account, and the containers do not have globally-unique naming requirements. Statistics are also tracked that include the total number of objects, and total storage usage for that container. The container services are more disk and network I/O intensive.

The Account Server is very similar to the Container Server, excepting that it is responsible for listings of containers rather than objects. Users can set metadata on their account, and swift aggregates usage information here. The account services are more disk and network I/O intensive.

2.7 Cloud Compute and Storage

Amazon Elastic MapReduce (Amazon EMR) [61] is a web service that enables to easily and cost-effectively process vast amounts of data. It utilizes a hosted Hadoop framework running on the infrastructure of Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Simple Storage Service (Amazon S3). It allows dynamic provisioning of storage capacity for performing data-intensive tasks. It releases the user from setting up, management or tuning of Hadoop clusters or the compute capacity upon which they reside.

Apache Hadoop [59] is an open-source software framework that supports data-intensive distributed applications, licensed under the Apache v2 license. It supports the running of applications on large clusters of commodity hardware. Hadoop is designed to run on normal hardware, provide fault-tolerance, deployability on low-cost hardware, serve big data. It uses parallelization approach and divides individual files into chunks sent to different computers. Its scale is hundreds or thousands of servers. It uses a master-slave or primary-secondary architecture, and there is a cluster single master named NameNode.

OpenStack Savanna [60] project attempts to enable users to easily provision and manage Hadoop clusters on OpenStack. Amazon provides Hadoop for several years as Amazon Elastic MapReduce (EMR) service. Savanna aims to provide users with simple means to provision a Hadoop cluster by specifying several parameters like Hadoop version, cluster topology, nodes hardware details. Savanna quickly deploys the cluster and provides means to scale already provisioned cluster by adding/removing worker nodes on demand. It is designed as an OpenStack component managed through REST API. Hortonworks [61] Mirantis and Red Hat are contributing engineering resources to Project Savanna.

The above technologies typically bring the data to the compute nodes that reside in the same data center. The required computations are performed by utilizing the CPU cycles of the compute nodes. In contrast, in PDS and the Storlet Engine, the relevant computations are performed (close to the data) utilizing the CPU cycles of the storage nodes.

3 Preservation DataStores (PDS) in the Cloud

3.1 What is PDS

PDS is an OAIS-based advanced preservation aware storage system that aims to provide a robust infrastructure for LTDP systems. The first version was designed and built by IBM for the CASPAR EU project in 2007. Then, it was extended for clouds environments in the ENSURE EU project that started in 2011, and the extended technology is called PDS Cloud. The terms PDS and PDS Cloud are used alternately to describe the full technology.

PDS Cloud provides OAIS support on top of generic clouds, and can operate over multiple diverse clouds. The figure below gives a high-level overview of the PDS Cloud. At the top, PDS Cloud provides an OAIS-based interface for operations on AIPs (e.g., ingest, access, delete), as well as an interface for preservation actions (e.g., check fixity, transform, add aggregation). At the bottom end, it utilizes various generic cloud storage and compute from different providers. In addition, the system includes a Storlet Engine that can be plugged into a private cloud or object storage to execute computation modules (called storlets) close to the data.

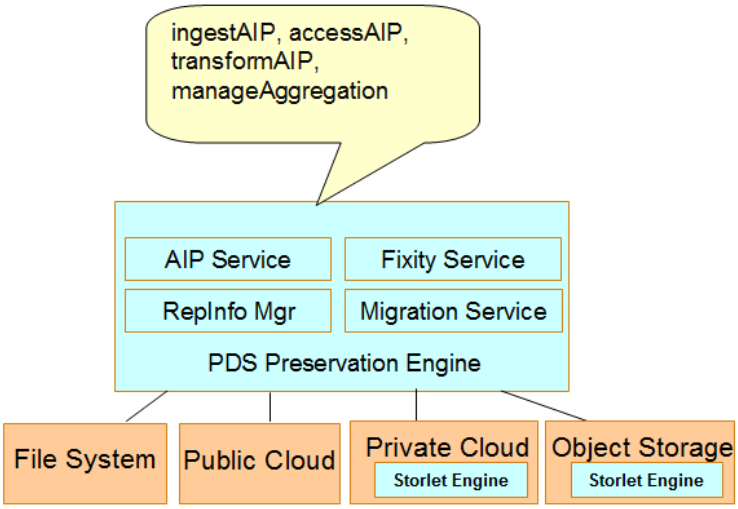

Figure 5: PDS Cloud system overview

PDS Cloud transforms the logical OAIS AIP information object into physical storage object(s). It performs preservation-related computations functions within the storage system via storlets (see section below for explanation and details on storlets).

3.2 PDS in CASPAR

PDS first version was built in CASPAR on top of an Object Storage Device (OSD) [21]. It is composed of a layered architecture, and employs open standards such as OAIS, XAM and OSD. It provides strong encapsulation of large quantities of metadata with the data at the storage level.

PDS in CASPAR is aware of the AIP structure and handles managing its sections. These functions include adding and managing RepInfo and validating their referential integrity, tracking provenance events, periodical computing of fixity, and performing migration. Moreover, PDS can perform transformation on the AIPs, relieving applications from data and I/O intensive tasks.

PDS supports basic AIP functions such as ingest (store) and access (retrieve), as well as setting/getting AIP metadata information, AIP versioning and unique identification. It includes a PDI Manager that handles PDI metadata-related actions (e.g., AIP provenance records, reference integrity) and a RepInfo Manager that manages all RepInfo related activities (e.g., maintaining a RepInfo repository, adding new RepInfos).

PDS implements and supports the CASPAR OAIS compliant authenticity model that includes authenticity protocols and steps [52]. It evaluates authenticity without application intervention or increase in network traffic, and avoids the associated security risks.

Currently, PDS does not support querying the provenance contents.

3.3 PDS in ENSURE

Cloud storage services provide opportunities to support novel models for long term preservation of digital assets, leveraging the cloud's inherent scalability and redundancy to dynamically adapt to evolving needs of digital preservation. The research on PDS Cloud was performed as the storage infrastructure component of the ENSURE EU project (Enabling kNowledge, Sustainability, Usability and Recovery for Economic Value) [2].

PDS Cloud [3] [63] is an OAIS-based preservation aware storage service that engages storage and compute clouds from diverse providers. It is a subsequent extended version of the OAIS-based PDS used in CASPAR that operates in a multi-cloud environment. It stores the AIPs in (possibly a variety of several) storage clouds via APIs provided by jclouds [7]. PDS Cloud utilizes the synergy of storage cloud and compute cloud. It introduced the notion of Virtual Appliances (see next subsection) that provide a compute/storage cloud synergy.

3.4 PDS extensions for the cloud

The gap analysis of existing storage clouds revealed that simply storing data onto the cloud is not an adequate solution for digital preservation repositories. PDS Cloud is designed to overcome some of these gaps. Extending PDS, PDS Cloud inherited the logical preservation and basic supported LTDP operations, and materializes the logical concept of a preservation information-object into a physical storage object. However, the physical storage object is kept in a storage cloud.
PDS Cloud offloads to the storage some preservation-related computations by using storlets. Note, however, that supporting storlets over a storage cloud entails enhancements to the storage cloud platform (e.g., enhancement for OpenStack Swift).

The additions of PDS Cloud (over PDS) address new cloud-specific goals and features. The main additions are:

1) Support access to multiple cloud storage and cloud compute platforms, as well as enable migration of data between different clouds.
2) Provide a flexible data model for multi-tenant multi-cloud environments.
3) Enhance future understandability of content by supporting data access using cloud based virtual appliances (VAs).
4) Offer advanced OAIS-based services in the cloud, such as fixity (integrity) checks, provenance and auditing that complement the generic clouds capabilities.
5) Provides a Storlets Engine (pluggable modules in the storage cloud) running storlets in a sandbox within the storage cloud.

3.5 Virtual Appliances (VAs)

Emulation techniques for interpretation and visualization of digital contents have been used to assist in logical preservation [32][33][34][35][36][37]. Yet, developing and maintaining emulation tools in each environment and for each information type may be too expensive to be a common solution for long term preservation.

Our approach advocates using an alternative that leverages the compute cloud instead of using specialized emulation systems. A virtual appliance (VA) is a virtual machine image (VM image) with an operating system and application packaged together as a pre-installed system image for a virtualized environment (e.g., KVM, Xen, VMware). The user runs the VA on a compute cloud (e.g., Amazon EC2, Rackspace Cloud Servers or OpenStack Nova) whereas the data resides on a storage cloud. PDS Cloud leverages a storage/compute cloud synergy to automate the provisioning of VAs.

The preserved information can be interpreted by the deployed VAs that hold the designated rendering software. VAs are thus an attractive alternative to specialized emulation environments. They are maintained transparent to regular users.

VAs are created by ingesting them into the PDS Cloud system as an OAIS AIP, which is designated (via special RepInfo) with the role of VA. VA Security is important, since PDS Cloud must guard against malicious intruders abusing the power of VAs, or legitimate users overreaching their privileges and accessing data that they have no permission to access. To this end, the current PDS Cloud solution limits the number of ports that accept incoming traffic.

3.6 Data Model

Enterprises using an archiving storage service typically organize their data in multiple collections having different defined policies and facilities for their data management, based on criteria such as information type, value to the organization, and storage cost. As the needs of the organization evolve over time, the data management profile of a collection should be dynamically configurable. Also, as cloud technology advances, it should be possible to easily migrate data to another cloud platform and leverage new cloud services. Ideally, any changes in data management should be completely transparent to the user of the storage service. Users should be able to access the data without needing to know underlying details such as the identity of the storage cloud providers, and should not have to adapt continuously to changes in configuration.

PDS Cloud data model enables transparent and dynamic configuration of data management in a multi-cloud multi-tenancy environment. The top-down data model hierarchy consists of: tenants, aggregations, dockets, and AIPs (see Figure below).

Tenant is an enterprise or organization that engages the preservation service. Each tenant has its own administrative ownership, regulations and users. For example in a personal use case, the tenant can be the person himself e.g. Peter Steiner.

Docket is a grouping of AIPs analogous to a directory in a filesystem. A docket contains zero or more AIPs, and has well-defined preservation metadata (key-value pairs). Each docket name is unique within the scope of a tenant. For example in a personal use case, the docket can be a specific trip of the person such as a trip to Costa Rica or a trip to Edinburgh.

Aggregation is a preservation profile for AIPs. It specifies details of the cloud platforms being used (address, credentials, etc.), and defines properties such as fixity (number and type of fixity modules, frequency of checks), etc. Every AIP instance is associated with (or belongs in) a specific aggregation. Different versions of the same AIP may be associated with different aggregations. The AIPs that belong in a given aggregation can be viewed as a collection of preservation objects that share the same preservation characteristics and are managed together. For example in a personal use case, we can have an aggregation of family photos that are very important to preserve (gold), an aggregation of business photos that are less important to preserve (silver), and hobby photos that are the least (bronze). The preservation value as computed by the managed forgetting process (see WP3) can be used to support this categorization into the different aggregations.

AIP (Archival Information Package) is the fundamental preserved entity in PDS Cloud. The typical AIP contains data in the application domain of the tenant.

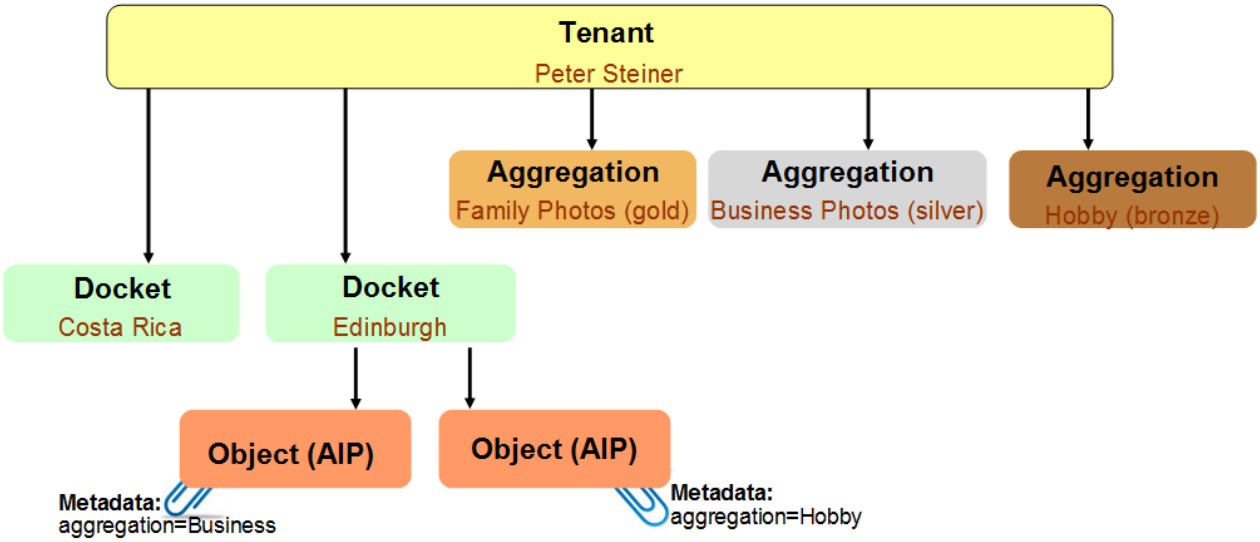

Figure 6: PDS Cloud Data model

3.7 Storlets

Offloading data maintenance functions to the storage system is common in many advanced storage systems (e.g., data migration, block level integrity, encryption). PDS was the first work to introduce the term storlets to denote general purpose (yet restricted) routines that may execute computations on the data close to the data within the storage layer (e.g., data transformations). Running within the storage close to the data, storlets can reduce the bandwidth required to move bytes to an application (possibly over WAN) for processing. It also improves the security and reduces the exposure of private data over the network. Storlets provide added flexibility, and consolidate logic that may otherwise require use of several applications. Storlets also simplify the LTDP system, which in turn results in higher overall system performance and reliability.

For example, a digital preservation system may need to perform periodical data intensive tasks, such as periodical validation and data transformation. These may be performed by predefined storlets. A predefined storlet itself (e.g., a fixity check script employing multiple algorithms) is stored in PDS as an AIP, whose content is the executable to be run (e.g., an operating system script).

3.8 PDS Cloud Architecture

The PDS Cloud architecture is depicted in the figure below. It constitutes a cloud broker that interconnects between the OAIS functions and the multiple diverse clouds (lower layer). It uses an existing multi-cloud interface service jclouds [7].

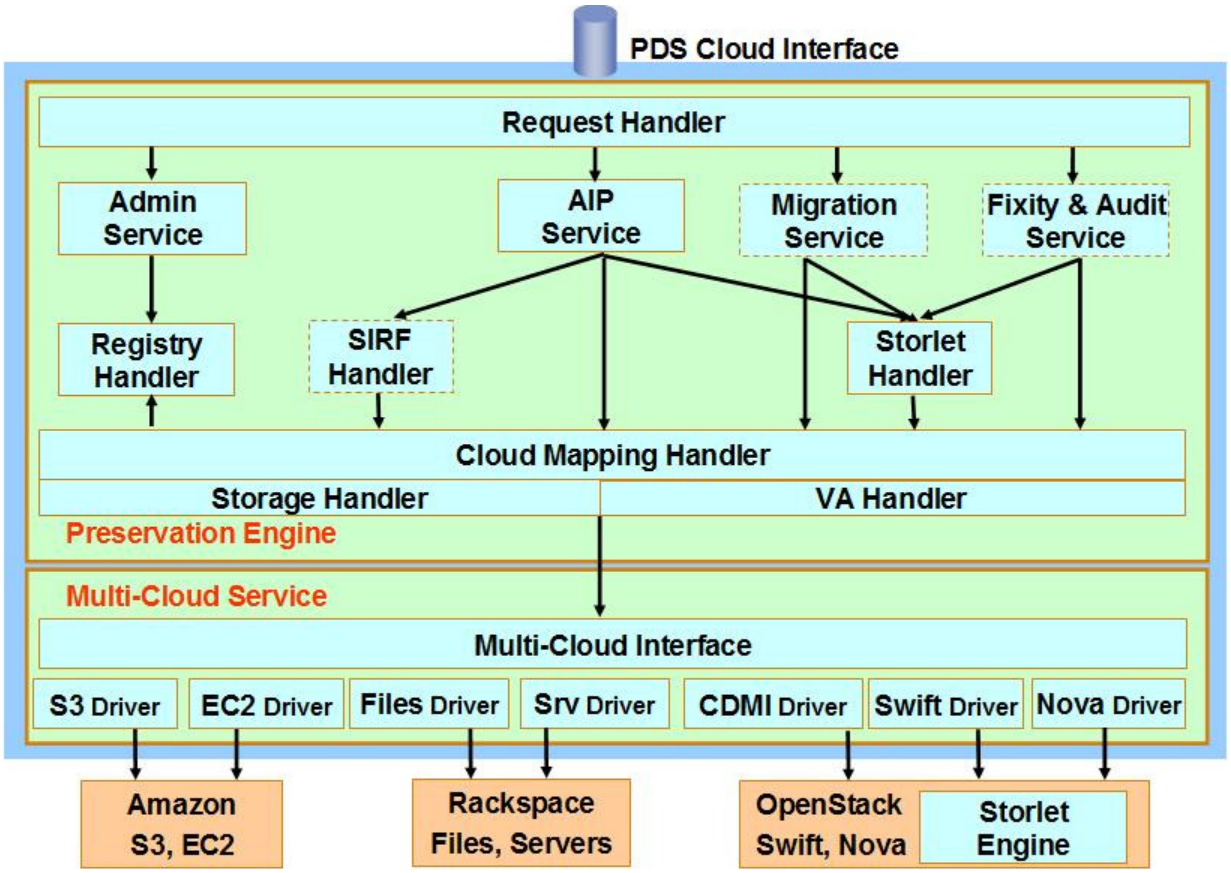

Figure 7: PDS Cloud architecture

PDS Cloud exposes to the client a set of OAIS-based preservation services such as ingest, access, delete of OAIS AIPs. It is assumed that user authentication and authorization to the preservation system are performed in the upper levels of the runtime environment before reaching PDS Cloud.

The preservation engine performs all the preservation related functionality. It is composed of several handlers: the Request Handler (HTTP server protocol parses, validate), the Cloud Mapping Handler (from AIPs to cloud objects), the Storage Handler and the VA Handler.

The core of the preservation engine is comprised by several services: a) AIP Service: ingest, access and delete of various types of AIPs (data, RepInfo, etc.) and metadata including provenance and relations among the various AIPs, b) Admin Service: tenants management, aggregations and policies, c) Migration Service, (future implementation), including transformations of AIPs from one format to another. (According to OAIS, migration comprises four functions: refreshment, replication, repackaging and transformation. The first three concern bit preservation, whereas transformation is a logical preservation operation), d) Fixity and Audit Service: flexible periodic integrity checks on AIPs using a choice of algorithms.

When the AIP Service, Migration Service and Fixity Service require performing data-intensive computational tasks, (such as validation, transformation etc.) they may use storlets. This requires extending the standard cloud storage capabilities, as current cloud platforms do not support storlets yet.

3.9 PDS is Extensible

PDS may be used as the storage subsystem in other preservation settings. The original implementation was built on top of an Object Storage Device, and another PDS variant was built on top of a plain file system (via IBM FileNet content management solution). Other existing solutions such as e-depot digital archiving

system of the National Library of the Netherlands (KB) [44] or the LOCKSS [48] and PASS [45] systems, may, in principle, be integrated with PDS to provide a more comprehensive LTDP solution.

Given its extensibility and flexibility, PDS and its code are good candidates to be leveraged in the ForgetIT framework. It has the potential (with proper extensions as presented below) to support the ForgetIT requirements from computational storage services.

4 Storlet Engine

4.1 Overview

Cloud storage utilizes server-based storage nodes with powerful CPUs to serve large data sets accessed from anywhere over the WAN. Taking it to the next step, we would like to exploit the storage node processing capabilities to execute restricted computational modules (storlets) within the data residency. The Storlet Engine provides the storage cloud with capability to run storlets in a sandbox close to the data. It provides a powerful extension mechanism to the cloud storage without changing its code; thus making the storage flexible, customizable and extensible. At a high level one can say that storlets in cloud storage are analogous to stored procedures in a database. In preservation systems, the storlet is maintained as an AIP, since it needs in itself to be preserved.

The benefits of using storlets are:

- Reduce bandwidth – reduce the number of bytes transferred over the WAN. Instead of moving the data over the WAN to the computational module, we move the computational module to the storage close to the data. Generally, the computational module has much smaller byte size than the data it works on.
- Enhance security – reduce exposure of sensitive data. Instead of moving data that has Personally Identifiable Information (PII) outside its residence, perform the computation in the storage and thereby lower the exposure of PII. This provides additional guard to the data and enables security enforcement at the storage level.
- Consolidate logic – expose generic functions that can be used by many applications. Instead of multiple applications writing similar code, storlets can consolidate and centralize generic logic with extensive or complex processing, and all applications can call these storlets. Additionally, this saves building computing infrastructure at the client side.
- Increase soundness – provide access to the latest preserved sound data. In preservation systems, AIPs may have multiple versions created over time, and the storage that manages all these versions has the up-to-date information about all versions and their relationship. Additionally, sound provenance of the data can be maintained as the actions on the data are done locally.

4.2 Storlet Composition and Sandbox

A storlet is composed of three parts:

- Lifecycle management – includes operations such as storlet deployment, storlet configuration and initialization, processes and threads management, storlet execution, inter-storlet communication. Lifecycle management is provided by the storlet engine.
- Business logic – the actual functionality of the storlet. This is provided by the application.
- Services invocation – calls performed by the storlet to external functionality provided by the storlet engine, e.g. calls to access objects in cloud storage.

A storlet may originate from several sources: written by a system administrator; written by a user; or bought as part of a third-party package or downloaded from some site. Further, storlet execution can be initiated at different privilege levels: by an administrator; by a privileged user; or by a regular user. Based on the source of the storlet, the initiator, and the storlet functionality, a certain level of trust should be associated with the storlet. Thus, the storlet runs in a sandbox that controls its execution and ensures that storlets can only perform actions that do not interfere with the performance and scalability of the storage system.

We provide two sandbox types to associate different levels of trust with different storlets:

- Admin Sandbox – the storlet can perform all operations.
- User Sandbox – the storlet is restricted and cannot perform administrative operations like writes on the file systems.

While in general system storlets will probably be assigned with admin sandboxes and application storlets assigned with user sandboxes, this is not always the case. System storlets can be assigned with user sandboxes and likewise privileged applications storlets can be assigned with admin sandboxes.

We started with storlets in the Java environment, since Java provides built-in support for restricted modules - servlets. While Java performance is sometimes considered as an issue, many optimizations have improved the performance of the JVM over time and it's becoming close to native code performance. The rest of this discussion will assume Java based storlets. Thus, a storlet is a servlet and the storlet developer may use the extensive existing development tools for servlets. The cloud platform initially targeted is OpenStack Swift.

4.3 Storlet Engine Operation

The Storlet Engine provides the cloud storage or object storage with capabilities to use storlets that run in a sandbox close to the data. It provides a powerful extension mechanism that makes the storage flexible, customizable and extensible. The engine is agnostic to the PDS Cloud data model and any preservation functionality. Thus, it can work with any application and any object that requires the storlets mechanism. Within the Storlet Engine the data model is the one used by the cloud storage and comprised of tenants, containers and objects.

The Storlet Engine supports two modes of operations:

- Synchronous mode – the storlet runs within the HTTP request/response that initiated its execution, or in other words the HTTP request ends after the storlet ends its execution. It can be synchronous on put if the request was an HTTP PUT request, synchronous on get if the request was an HTTP GET request, etc. We also call this streaming mode.
- Asynchronous mode – the HTTP request that initiated the storlet execution ends as soon as the system registered the request. The storlet runs in the background and may accumulate results in an output object that also includes information about its completion status. The storlet initiator may later on access the output object and retrieve the results of the computation. We also call this batch mode.

The following figure depicts how storlets can be used for example from the OpenStack dashboard.

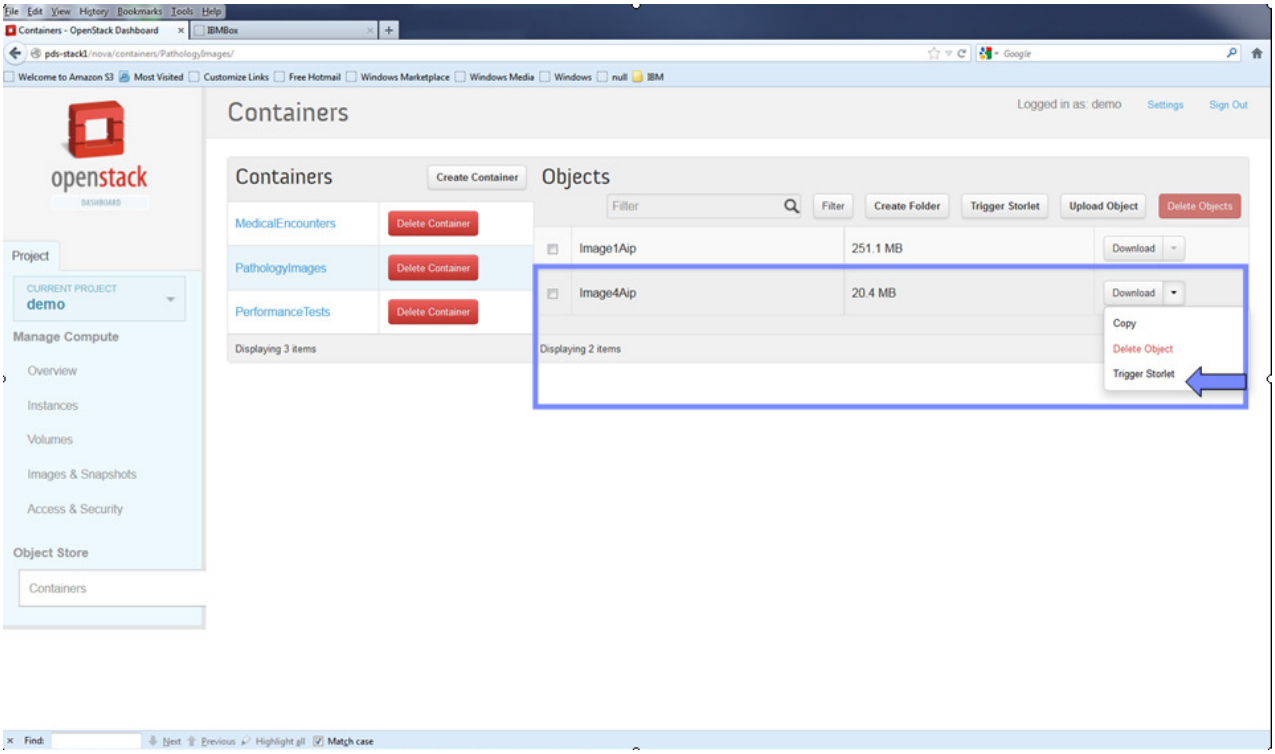

Figure 8: Trigger Storlet in OpenStack Dashboard

Taken together, The Storlet Engine provides a mechanism to leverage the computational power available at the storage server nodes level to process data. It is especially beneficial for processing large data sets, which alternatively would have to be transferred over the network to an application server. Typically, storlets should be limited to relatively simple, yet data intensive operations.

5 Initial Design Directions for ForgetIT Work Package 7

5.1 Addressing the Goals

Based on the review of the foundations of computational storage presented above, we wish to build a preservation aware LTDP storage system that will meet the ForgetIT requirements and goals, and demonstrate its functionality on the ForgetIT use cases. Hence, we plan to design a consolidated object and compute architecture for storage objects and computational processes (Storlet Engine) that may be defined, triggered and executed close to the data, in a cloud environment.

5.2 Our Starting point is PDS and OpenStack Swift

We wish to leverage existing technology as much as possible. The selected base LTDP system we chose is PDS Cloud, and the selected cloud platform is the existing OpenStack/Swift. These will require extensions and refinements to specifically support the ForgetIT framework and goals. We will define the structure of the AIPs for ForgetIT data, and implement the creation of the AIPs from the raw data and metadata. The major extension will be enhancements to the existing PDS Cloud storlet mechanism. Storlets will be packaged as AIPs, and will be invoked via a REST HTTP request using a well defined format.

5.2.1 The planned Storlet extensions

Storlets may be extended as follows:
a) Executed in a sand-box and include specialized security model.
The guarantee of enhanced security storlets will promote the adoption of preservation related storlets by potential security sensitive users, in support of the ForgetIT goals.
b) Added/removed dynamically
The possibility to easily add/remove storlets will promote the adoption of preservation related storlets, and will enable evolving managed forgetting and contextualized remembering.
c) Input/output to storlets may be multiple objects (also objects may be versioned)
This feature may be leveraged by dynamic evolution-aware contextualization, which require simultaneous processing of many objects.
d) Support for streaming operations, batch operations, and periodic operations
This feature is useful for managed forgetting and synergetic preservation.
e) Some metadata support for storlets
Storlets metadata is important for allowing storlets to perform general purpose data processing activities.
f) Optimize storlet placement so it will run best and be closest to its data
This feature will optimize storlet performance and will promote its usage by potential users.
g) Parallel and distributed execution of storlets
This feature will optimize data intensive processing tasks in which the task may be broken into parallel independent sub tasks.

Examples of Potential Storlets for ForgetIT:
a) Summarization and aggregation processes to enable managed forgetting.
The storlet may scan objects to perform summarization per policy criteria.
b) Redundancy detection and deletion processes to enable managed forgetting.
The storlet may scan objects to perform removal of redundancy or complete removal per policy criteria.
c) Multimedia analysis algorithms (provided by CERTH in work package 4)
The storlet may efficiently perform image analyses at the storage server level close to the data.
d) Analytics processes
The storlet may efficiently perform analytics processing at the storage server level close to the data.
e) Indexing processes
The storlet may efficiently perform indexing processing at the storage server level close to the data. This time consuming processing may be performed at the storage server level as a background process, without loading the application servers.
f) Encryption/Secure Delete

The storlet may perform encryption of a copy of an object, and optionally delete the original clear-text version.
g) Format transformations
The storlet may perform format transformations to enhance object readability, save storage space, conform to a standard and/or improved logical preservation for future uses.

Potential storlets developed jointly with CERTH partner:
(1) Image feature extraction
(2) Image concept detection

Potential storlets developed jointly with Turk Telekom partner:
(1) Virus check - can be useful as a storlet if we wish to dynamically upload new checkers, possibly from new vendors and scan data that is already stored.
(2) Video transcoding – transcode for mobile devices e.g. with VideoLAN
(3) Document conversion – convert from Word to PDF or JPEG. This is useful for automatically converting proprietary formatted documents (e.g., Word format) to non-proprietary formats (e.g., JPEG) for viewing with general purpose applications.

5.3 Additional Possible Directions

One possible additional direction is to allow for some form of (limited) searchable metadata. A search is desired for our ForgetIT variant to support listing of objects on which to perform preservation actions.

Another possible direction is to implement support for SIRF (Self-contained Information Retention Format), to support SIRF containers in the cloud. SIRF provides a catalog with metadata related to the entire contents of the container as well as to the individual objects and their interrelationship.

5.4 Planned Integration with the "Preserve-or-Forget" Framework

The planned integration will be via HTTP RESTful requests. This forms a loose form of integration, which guarantees simplicity and future portability. It will also allow for easy testing and smooth integration. The details of the integration will be defined in D8.1.

5.5 Other Possible Directions

As time permits, we may explore additional extension directions. Examples include: exchange with external cloud (e.g., DSpace), addressing joint information management and preservation lifecycle via development of a reference model, smooth bi-directional preservation/information management storage transitions, indexing methods and analytics.

6 Summary and Conclusions

We have presented a review of the foundations of computational storage, in the context of the ForgetIT project and goals, with a focus on the storage system and LTDP. The general goal is to increase the value and outcome of preserved information over time and provide additional incentive to the potential users for preservation by transforming the generic storage service to a richer service with potentially higher business and personal value.

Based on the state of the art, we will build a consolidated platform for objects and computational processes (Storlet Engine) that is defined, triggered and executed close to the data.

We will leverage existing storage technology (PDS Cloud) and cloud technology (OpenStack) and extend both as needed to achieve the planned ForgetIT goals and produce the expected ForgetIT results, demonstrated with some of ForgetIT use cases data and computations.

References

[1] Reference Model for an Open Archival Information System (OAIS) - Recommended Practice, CCSDS 650.0-M-2 (Magenta Book) Issue 2. Also available as ISO Standard 14721:2012. The Consultative Committee for Space Data Systems (CCSDS), June 2012.

[2] ENSURE: Enabling kNowledge Sustainability, Usability and Recovery for Economic value, EU FP7 project. URL http://ensure-fp7.eu.

[3] S. Rabinovici-Cohen, J. Marberg, and K. Nagin. Preservation DataStores in the Cloud (PDS Cloud): Long term digital preservation in the cloud. Technical Report H0318, IBM Research Haifa, January 2013.

[4] S. Rabinovici-Cohen, M. Factor, D. Naor, L. Ramati, P. Reshef, S. Ronen, J. Satran, and D. Giaretta. Preservation DataStores: New storage paradigm for preservation environments. IBM Journal of Research and Development, Special Issue on Storage Technologies and Systems, 52(4/5):389-399, July/September 2008.

[5] Amazon Web Services. URL http://aws.amazon.com.

[6] Openstack cloud software. URL http://openstack.org.

[7] Jclouds. URL http://www.jclouds.org.

[8] Edmund B. Nightingale, Jeremy Elson, Jinliang Fan, Owen Hofmann, Jon Howell, and Yutaka Suzue, "Flat Datacenter Storage", in 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 2012). http://research.microsoft.com/apps/pubs/default.aspx?id=170248

[9] E.K. Kolodner, S. Tal, D. Kyriazis, D. Naor, M. Allalouf, L. Bonelli, P. Brand, A. Eckert, Elmroth E, S.V. Gogouvitis, Harnik D, F. Hernandez, M.C. Jaeger, E.B.Lakew, J.M. Lopez, M. Lorenz, A. Messina, A. Shulman-Peleg, R. Talyansky, A. Voulodimos, and Y. Wolfsthal. A cloud environment for data-intensive storage services. In Cloud-Com 2011: Proceedings of the IEEE Third Inter-national Conference on Cloud Computing Technol-ogy and Science, pages 357{366, Athens, Greece, November 2011. URL http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6133164.

[10] Eucalyptus. URL http://www.eucalyptus.com.

[11] The Rackspace Cloud. URL http://www.rackspace.com/cloud.

[12] EMC Atmos. URL http://www.emc.com/storage/atmos/atmos.

[13] DuraCloud. URL http://www.duracloud.org.

[14] L. Richardrson and S. Ruby. RESTful Web Services. O'Reilly Media, 2007. URL http://shop. oreilly.com/product/9780596529260.do.

[15] S. Rabinovici-Cohen, M. Factor, D. Naor, L. Ramati, P. Reshef, S. Ronen, J. Satran, and D. Giaretta. Preservation DataStores: New storage paradigm for preservation environments. IBM Journal of Research and Development, Special Issue on Storage Technologies and Systems, 52(4/5):389{399, July/September 2008. URL

[16] http://www.research.ibm.com/haifa/projects/storage/datastores/papers/rabinovici.pdf.

[17] Y. Li, D.D.E. Long, and E.L. Miller. Understanding data survivability in archival storage systems. In SYSTOR 2012: Proceedings of the 5th Annual International Systems and Storage Conference, Haifa, Israel, June 2012. URL http://www.soe.ucsc.edu/~yanli/res/li-systor12.pdf.

[18] P. Mell and T. Grance. The NIST definition of cloud computing: Recommendations of the National Institute of Standards and Technology. Special Publication 800-145, National Institute of Standards and Technology, U.S. Department of Commerce, September 2011. URL http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf.

[19] Cloud Data Management Interface (CDMI), Version 1.0.2, Technical Position. Storage Networking Industry Association (SNIA), June 2012. URL http://cdmi.sniacloud.com.

[20] S. Rabinovici-Cohen, M.G. Baker, R. Cummings, S. Fineberg, and J. Marberg. Towards SIRF: Self-contained Information Retention Format. In SYSTOR 2011: Proceedings of the 4th Annual International Systems and Storage Conference, Haifa, Israel, May 2011. URL http: //www.research.ibm.com/haifa/projects/storage/datastores/papers/systor56-rabinovici-cohen.pdf.

[21] M. Factor, D. Naor, S. Rabinovici-Cohen, L. Ramati, P. Reshef, J. Satran, and D. Giaretta. Preservation DataStores: Architecture for preservation aware storage. In MSST 2007: Proceedings of the 24th IEEE Conference on Mass Storage Systems and Technologies, pages 3-15, San Diego, CA, September 2007. URL http://www.haifa.il.ibm.com/projects/storage/datastores/papers/Preservation DataStores MSST07 camera.pdf.

[22] L. Sweeney. Achieving k-anonymity privacy protection using generalization and suppression. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 10(5):571{588, October 2002. URL http://www.comp.nus.edu.sg/~tankl/cs5322/readings/k-anonymity2.pdf.

[23] Tradestation. URL http://www.tradestation.com.

[24] M. Baker, K. Keeton, and S. Martin. Why traditional storage systems don't help us save stufforever. In HotDep 2005: Proceedings of the IEEE First Workshop on Hot Topics in System Dependability, Yokohama, Japan, June 2005. URL http: //www.hotdep.org/2005/baker forever.pdf.

[25] M. Baker, M. Shah, D. Rosenthak, M. Roussopoulos, P. Maniatis, T.J. Giuli, and P. Bungale. A fresh look at the reliability of long-term digital storage. In EuroSys 2006: Proceedings of the 1st ACM SIGOPS European Systems Conference, pages 221{234, April 2006. URL http://dl.acm.org/citation.cfm?id=1141770.

[26] M. Factor, D. Naor, S. Rabinovici-Cohen, L. Ramati, P. Reshef, and J. Satran. The need for preservation aware storage - a position paper.ACM SIGOPS Operating Systems Review, Special Issue on File and Storage Systems, 41(1):19{23, January 2007. URL http://www.research.ibm.com/haifa/projects/storage/datastores/papers/preservation data store osr07 dec 30.pdf.

[27] A. Dappert and M. Enders. Digital preservation metadata standards. Information Standards Quarterly, Special Issue on Digital Preservation, 22(2):4{12, Spring 2010. URL http://www.loc.gov/standards/premis/FE Dappert Enders MetadataStds isqv22no2.pdf.

[28] K-K Muniswamy-Reddy, P. Macko, and M.I. Seltzer. Provenance for the cloud. In FAST 2010: Proceedings of the 8th USENIX Conference on File and Storage Technologies, pages 197{210, San Jose, CA, February 2010. URL http://www.usenix.org/events/fast10/tech/full papers/muniswamy-reddy.pdf.

[29] Digital Archive. URL http://www.oclc.org/us/en/digitalarchive.

[30] A. Bessani, M. Correia, B. Quaresma, F. Andre, and P. Sousa. Depsky: Dependable and secure storage in a cloud-of-clouds. In EuroSys'11: Proceedings of the 6th ACM EuroSys Conference on Computer Systems, pages 31{46, Salzburg, Ausria, April 2011. URL http://www.gsd.inesc-id.pt/~mpc/pubs/eurosys219-bessani.pdf.

[31] L.L. You, K.T. Pollack, D.D.E. Long, and K. Gopinath. PRESIDIO: A framework for e-cient archival data storage. ACM Transactions on Storage, 7(2), July 2011. URL http://doi.acm.org/10.1145/1970348.1970351.

[32] J.R. Van der Hoeven, H.N. van Wijngaarden, R. Verdegem, and J. Slats. Emulation - a viable preservation strategy. Technical report, Koninklijke Bibliotheek / Nationaal Archief, The Hague, Netherlands, May 2005. URL www.kb.nl/hrd/dd/dd projecten/Emulation research KB NA 2005.pdf.

[33] J.R. Van der Hoeven and H.N. van Wijngaarden. Modular emulation as a long-term preservation strategy for digital objects. In IWAW'05: Proceedings of the 5th International Web Archiving Workshop, Vienna, Austria, May 2005. URL http://iwaw.europarchive.org/05/papers/iwaw05-hoeven.pdf.

[34] R.A. Lorie. A methodology and system for preserving digital data. In JCDL 2002: Proceedings of the 2nd ACM/IEEE-CS Joint Conference on Digital Libraries, pages 312{319, Portland, OR, July 2002. URL http://doi.acm.org/10.1145/544220.544296.

[35] R.A. Lorie. The UVC: A method for preserving digital documents: Proof of concept. Technical report, Koninklijke Bibliotheek / Nationaal Archief, The Hague, Netherlands, 2004. URL http://www. kb.nl/hrd/dd/dd onderzoek/reports/4-uvc.pdf.

[36] T. Reichherzer and G. Brown. Quantifying software requirements for supporting archived once documents using emulation. In JCDL 2006: Proceedings of the 6th ACM/IEEE-CS Joint Conference on Digital Libraries, pages 86{94, Chapel Hill, NC, June 2006. URL http://dl.acm.org/citation. cfm?doid=1141753.1141770.

[37] D. Von Suchodoletz. A future emulation and automation research agenda. In Automation in Digital Preservation, Dagstuhl Seminar Proceedings 10291, Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2010. URL http: //drops.dagstuhl.de/opus/volltexte/2010/2771.

[38] S. Rhea, R. Cox, and A. Pesterev. Fast, inexpensive content-addressed storage in foundation. In USENIX'08: Proceedings of the USENIX 2008 Annual Technical Conference, pages 143{156, Boston, MA, June 2008. URL http://static.usenix.org/events/usenix08/tech/full papers/pucha/pucha.pdf.

[39] C.W. Bailey, Jr. Digital curation bibliography: Preservation and stewardship of scholarly works. Technical report, Digital Scholarship, Houston, TX, 2012. URL http://digital-scholarship.org/dcpb/dcb.htm.

[40] Michael Peterson, Gary Zasman, Peter Mojica, and Jeff Porter. 100 year archive requirements survey. Technical report, Storage Networking Industry Association, January 2007.

[41] Self-contained Information Retention Format (SIRF) use cases and functional requirements, working draft – version 0.5a, SNIA, September 2010, http://www.snia.org/tech_activities/publicreview/SIRF_Use_Cases_V05a_DRAFT.pdf

[42] Simona Rabinovici-Cohen, Mary G. Baker, Roger Cummings, Sam Fineberg, and John Marberg, "Towards SIRF: Self-contained Information Retention Format", Proceedings of the Annual International Systems and Storage Conference (SYSTOR), May 30-June 1, 2011, Haifa, Israel

[43] M. W. Storer, K.M. Greenan, E.L. Miller, K. Voruganti. "POTSHARDS: Secure Long-Term Storage Without Encryption". In Proceedings of the 2007 USENIX Technical Conference, June 2007

[44] Erik Oltmans, Raymond J. van Diessen, Hilde van Wijngaarden. "Preservation Functionality in a Digital Archive". Digital Libraries, 2004 ACM/IEEE Joint Conference on (JCDL'04), 2004, pp. 279-286. See http://www.kb.nl/e-depot. DIAS - Digital Information Archiving System. See http://www-5.ibm.com/nl/dias/preservation.html

[45] K. Muniswamy-Reddy, D.A. Holland, U. Braun, and M. Seltzer. "Provenance-aware storage systems". In Proceedings of the 2006 USENIX Annual Technical Conference. June 2006

[46] D. A. Holland, U. Braun, D. Maclean, K.K. Muniswamy-Reddy, and M. Seltzer. Choosing a Data Model and Query Language for Provenance. In proceedings of the 2nd International Provenance and Annotation Workshop, Salt Lake City, UT, Jun 2008

[47] Venti, see: http://www.stanford.edu/class/cs240/readings/venti-fast.pdf

[48] LOCKSS, see: http://library.stanford.edu/projects/lockss

[49] SRB, see: http://www.sdsc.edu/srb/index.php/Main_Page

[50] iRods, see: https://www.irods.org/index.php/Introduction_to_iRODS

[51] DeepStore, see: http://www.ssrc.ucsc.edu/pub/you05-icde.html

[52] "Authenticity and provenance in long term digital preservation: modeling and implementation in preservation aware storage", Michael Factor, Ealan Henis, Dalit Naor, Simona Rabinovici-Cohen, Petra Reshef, Shahar Ronen, Giovanni Michetti and Maria Guercio, The first workshop on the Theory and Practice of Provenance (TaPP 09), San Francisco, USA, February 2009. Also published in Journal Archivi & Computer Automazione e Beni Culturali, Vol. 2-3 2009, pp. 93-102

[53] S3, see: http://aws.amazon.com/s3/

[54] EC2, see: http://aws.amazon.com/ec2/

[55] OpenStack Swift, see: https://wiki.openstack.org/wiki/Swift

[56] OpenStack Nova, see: http://docs.openstack.org/developer/nova/

[57] Windows Azure, see: http://www.windowsazure.com/en-us/

[58] VISION Cloud, see: http://www-03.ibm.com/press/us/en/pressrelease/32996.wss

[59] Hadoop. See: http://en.wikipedia.org/wiki/Apache_Hadoop

[60] OpenStack Savanna, see: https://wiki.openstack.org/wiki/Savanna, https://wiki.openstack.org/wiki/Savanna/ProjectBlueprint, https://wiki.openstack.org/wiki/Savanna/Roadmap

[61] Hortonworks, see: http://hortonworks.com/about-us/news/hortonworks-mirantis-and-red-hat-team-to-simplify-deployment-and-management-of-apache-hadoop-on-openstack/

[62] MapReduce, see: http://aws.amazon.com/elasticmapreduce/

[63] S. Rabinovici-Cohen, J. Marberg, K. Nagin, and D. Pease. PDS Cloud: Long term digital preservation in the cloud. In IC2E 2013: Proceedings of the IEEE International Conference on Cloud Engineering, San Francisco, CA, March 2013.