Preservation DataStores: New storage paradigm for preservation environments

S. Rabinovici-Cohen
M. E. Factor
D. Naor
L. Ramati
P. Reshef
S. Ronen
J. Satran
D. L. Giaretta

As the world becomes digital, we are in ever greater danger of losing business, scientific, artistic, cultural, and personal assets. The threat of such a digital dark age stems from the fact that—unlike physical records that may survive decades, centuries, or even longer without advanced planning—digital records will not survive without planning and diligence. Everything needed to keep digital records viable will become obsolete, including hardware, software, processes, and formats. Consequently, digital preservation environments are needed to ensure the ability to access valuable digital records decades from now and, more significantly, to ensure the interpretability of the records once accessed. We describe Preservation DataStores, an innovative storage architecture that facilitates robust and optimized preservation environments. It is a layered architecture that builds upon open standards, including Open Archival Information System, XAM (Extensible Access Method), and Object-based Storage Device. We also describe the integration of Preservation DataStores with existing file systems and archives and discuss some design and implementation issues. We are developing Preservation DataStores as an infrastructure component of the European Union CASPAR (Cultural, Artistic and Scientific knowledge for Preservation, Access and Retrieval) project, where it will be used to preserve scientific, cultural, and artistic data.

Introduction

Today we are facing a paradox. We can read and interpret the Dead Sea scrolls created two millennia ago, but most of us no longer have the means to read or interpret data we may have generated ourselves two decades ago on a 5.25-inch floppy disk. Nevertheless, long-term preservation of digital data is being required by new regulations, such as the Health Insurance Portability and Accountability Act (HIPAA), Sarbanes–Oxley, Occupational and Safety Health Administration (OSHA) regulations, and other federal securities laws and regulations. These rules can require keeping medical data for the life of a patient or financial data for the life of an account. In addition to regulatory requirements, there are business, cultural, and personal needs to preserve digital information for periods longer than the lifetime of the technology that was used to create the data. For example, earth observation data from the European Space Agency and cultural heritage data from UNESCO (United Nations Educational, Scientific and Cultural Organization) must be kept for decades and centuries. Finally, the amount of long-lived data is expected to grow even more with the vast amounts of digital data being generated by emerging digital devices.

Digital preservation systems aim to ensure that long-lived data will be usable in the distant future. Digital preservation comprises two aspects: *bit preservation*, which is the ability to access the bits of the digital record,

0018-8646/08/$5.00 © 2008 IBM

and *logical preservation*, which is the ability to use and understand the data in the future. In addition, logical preservation must support tracking the provenance of a record and ensuring its authenticity and integrity. Bit preservation issues are mostly well understood and are supported by existing products used to migrate data between different generations of storage technologies. In contrast, logical preservation is still mostly an unsolved problem.

Open Archival Information System (OAIS) [1] is an ISO (International Organization for Standardization) standard for logical preservation approved in 2003 (ISO 14721:2003 OAIS). OAIS specifies information and functional models for preserving digital assets used by a community of users (designated community) from the moment the digital material is ingested into the digital storage area through subsequent preservation strategies to the creation of a dissemination package for the end user. (*Ingest* means accepting data and its relevant metadata and preparing it for storage and management within the archive.) OAIS is a high-level reference model and does not provide implementation guidance. The European Union (EU) CASPAR (Cultural, Artistic and Scientific knowledge for Preservation, Access and Retrieval) project is trying to make OAIS concrete. It is building a framework and architecture based on OAIS; CASPAR will be used to demonstrate the preservation of digital cultural, artistic, and scientific data [2].

At the heart of any solution for digital preservation resides a storage component that is responsible for the permanent location of the information. Traditional archival storage considers only bit preservation issues if it considers preservation at all. It provides functions to ingest and retrieve data from the permanent storage, manage a storage hierarchy, handle media degradation, perform integrity checks on the data, and support disaster recovery.

In Reference [3], we defined the new concept of preservation-aware storage that addresses both bit preservation and logical preservation requirements. We argued that offloading preservation-related functionality to the storage component will increase the robustness of digital preservation systems and their ability to protect the data stored on them against corruption or loss. By adding such functionality to the storage component, we reduce data transfers between this component and others and improve data placement in favor of preservation.

In this paper, we define Preservation DataStores (PDS), an architecture for preservation-aware storage, and describe its benefits. (A preliminary version of parts of this paper appeared in Reference [4].) We present design and implementation issues encountered while developing the PDS prototype. Finally, as data subject to long-term digital preservation may already reside in an existing file system or archive, the paper explains how PDS can be integrated with an existing repository. PDS is being developed as an infrastructure component of CASPAR.

CASPAR

CASPAR builds an OAIS-based framework for preservation of cultural, artistic, and scientific knowledge. CASPAR focuses on the challenging issue of logical preservation and preservation of the understandability of the information encoded in the digital objects. For example, one could make a digital object by carving 1s and 0s in stone, a very durable way to preserve information, as demonstrated by the ancient Egyptians. However, while this may give one access (slow access but nevertheless access), it will not maintain understandability, as shown by the Phaistos disk dated to 1700 BC, which has still not been translated [5].

CASPAR argues that no single organization, project, or person can guarantee funding, efforts, or even interest in the preservation of digital objects. Instead, a consortium must share the responsibility for preservation. Thus, the project brings together a consortium covering important digital holdings with the appropriate extensive scientific (lead partners Science and Technology Facilities Council U.K. and the European Space Agency), cultural (UNESCO), and creative arts expertise (Institut National de l'Audiovisuel, Centre National de la Recherche Scientifique, University of Leeds, Institut de Recherche et Coordination Acoustique/Musique, and the International Centre for Art and New Technologies), together with commercial partners (Advanced Computer Systems, Asemantics, Metaware, Engineering Ingegneria Informatica, and IBM Haifa Research), experts in knowledge engineering (Consiglio Nazionale delle Ricerche/Institute of Information Science and Technologies and the Foundation for Research and Technology), and academic partners working in this area (Universities of Glasgow and Urbino).

Enabling the transfer of holdings to the next in the chain of preservation involves more than just transferring the bit streams of individual objects; objects are often parts of one or more collections, each of which needs description. The objects need persistent identifiers by which users can find the information and there may be several levels of virtualization that need attention. CASPAR is attempting to enable these capabilities related to the storage elements as well as all the other types of preservation components including digital rights management and provenance that must be in place for the effective long-term preservation of digitally encoded information.

One of the difficult issues in preservation systems is their validation methods. How do you validate today that

your system actually preserves data for a long time when the future is unpredictable? A primary objective of the CASPAR project is to design such validation methods.

Related work

The purpose of this section is to survey archival storage systems and focus on their ability to address digital preservation issues. Although we are seeing an increasing focus on long-term issues, systems so far have addressed either the bit aspect of the preservation or the access aspect. In our view, the logical aspect has thus far been unexplored.

Storage aspects of the digital preservation problem have been the focus of a growing number of studies [6–8]. Long-term preservation systems differ from traditional storage applications in terms of goals, characteristics, threats, and requirements. Baker et al. [6] examine these differences and suggest guidelines for alternative architectural solutions that focus on replication across autonomous sites, reduced per-site engineering costs, and the ability to scale over time and technologies. Baker et al. [7] explore the need for long-term preservation systems and the threat to long-lived digital information. An extended reliability model is presented along with several strategies for reducing the probability of irrecoverable data loss. All of the studies mentioned above review different new directions and challenges in the storage aspect of long-term preservation while concentrating primarily on bit preservation.

The long-term digital preservation problem is further explored by Rodriguez [9] and Peterson [10], who provide motivation and requirements for a general solution. The National Digital Information Infrastructure and Preservation Program [11] aims to develop an overall national strategy for the preservation problem. The initiative is supervised by the Library of Congress. Its goal is to provide a high-level architectural framework and to define a set format that will comply with specific preservation needs.

Various security aspects are encompassed in the challenge of long-term preservation. Storer et al. [12] examine archive security threats and how they are addressed by existing systems. More specifically, the authors state that while several security threats are already known (e.g., integrity, authentication, and privacy), others are introduced due to the long-term aspect of the problem (e.g., slow attacks may span a long period of time, even several decades; the assumption is that the long period of time gives the attacker more opportunities to gain access to more information). The POTSHARDS (protection over time, securely harboring, and reliably distributing stuff) system [13] aims to provide secure long-term storage while shifting data secrecy from encryption to authentication by using secret splitting across multiple authentication domains.

The Storage Resource Broker (SRB) [8, 14, 15] developed by the San Diego Supercomputer Center is a logical distributed file system based on a client/server architecture that presents the user with a single global logical namespace or file hierarchy. SRB is a data grid technology that can be considered middleware since it is built on top of file systems, archives, real-time data sources, and storage systems. Several preservation projects, such as the research prototype for the National Archives and Records Administration, utilize the SRB as the foundation for their archives. The SRB code was recently extended to a new system called iRODS [16], Intelligent Rule-Oriented Data Management System. The iRODS rule engine can be useful for implementing preservation-specific policies.

Venti [17] 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, it provides the back-end archival storage for these types of applications. In the system, a 160-bit secure hash algorithm, SHA-1, 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 with the fingerprint held by the application.

Deep Store [18] is an 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 that includes PRESIDIO (Progressive Redundancy. Elimination of Similar and Identical Data in Objects), a new lossless data compression framework that incorporates interfile 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 (Extensible Markup Language) format, and methods for searching and versioning the metadata are provided.

Storage systems such as SRB, Venti, and Deep Store provide storage with characteristics customized for archival data. However, these systems do not take into account the complex preservation environment and provide only bit preservation, as opposed to our approach, which includes support for both bit and logical preservation.

Several projects have already adapted the OAIS model and developed technologies aimed at addressing the

overall preservation problem. The e-depot digital archiving system of the National Library of the Netherlands is described by Oltmans et al. [19]. The e-depot library conforms to the OAIS standard and supports physical and logical digital preservation. The e-depot is based on the IBM Digital Information Archiving System (DIAS), which provides a flexible and scalable open deposit library solution for storing and retrieving massive amounts of electronic documents and multimedia files. LOCKSS (Lots of Copies Keep Stuff Safe) [20] is an OAIS-compliant open-source, peer-to-peer decentralized digital preservation infrastructure. The project, which originated at Stanford University, provides software that turns an ordinary personal computer into a digital preservation appliance and implements a protocol for auditing and repairing the online content. The e-depot and LOCKSS projects aim to present a general framework to the preservation problem. They support such aspects as bit degradation and format obsolescence. Our solution, PDS, is used as the storage archive component of the CASPAR project and may be utilized in any of the above preservation solutions.

Offloading functionality from the application to the storage system is an emerging trend. Functionalities such as bit-to-bit data migration, block-level data integrity, and encryption are now carried out by advanced intelligent disks and tapes. Provenance-aware storage systems, known as *PASS* [21], track the provenance of data at the storage level, while other systems manage provenance tracking in a standalone database separate from the data. Our solution provides built-in support for various aspects of the preservation process, such as fixity computation (computations to verify the integrity of the data) and provenance tracking.

Several public projects were launched recently and their main focus is to enable the sharing and management of digital assets. The Preservation and Long-term Access through Networked Services (PLANETS) [22] project aims to evaluate the different preservation strategies and provide services and tools to help ensure long-term access to digital cultural and scientific assets. The Building Resources for Integrated Cultural Knowledge Services (BRICKS) [23] project investigates and implements advanced open-source software solutions based on standards for the sharing and exploitation of digital cultural resources. The goal of the Digital Library Infrastructure on Grid-Enabled Technology (DILIGENT) [24] project is to integrate grid and digital library technologies. It aims to create an advanced test bed that will enable virtual e-science communities to collaborate and share knowledge.

Preservation DataStores

An important part of any preservation system is its storage component, which is responsible for the long-term storage and maintenance of digital material. Preservation-aware storage is a storage component with built-in support for preservation. Preservation-aware storage encapsulates the record to be preserved with the metadata required for managing future access and use of this record. It enables offloading data-intensive preservation functions and storage-related responsibilities from the application to the storage. This decreases the number of data transfers and simplifies applications. Preservation-aware storage performs fixity computations, migrations, and externally specified logical transformations within the storage. It also internally manages storage-related events, such as provenance events.

PDS is an OAIS-based, preservation-aware storage with the main goal of supporting logical preservation. The idea behind PDS is to transform the logical information object, a basic concept in preservation systems, into physical storage objects. By doing this, PDS can perform preservation-related functions within the storage, thus providing a robust and optimized implementation for preservation-aware storage. PDS is composed of a layered architecture that complies with the general design principle of preservation systems and employs open standards wherever possible; among these standards are OAIS, XAM (Extensible Access Method) [25], and Object-based Storage Device (OSD) [26].

To provide a better description of PDS, we start by summarizing some relevant OAIS concepts. A fundamental OAIS concept utilized in PDS is the archival information package (AIP), which is the main OAIS logical information object used in the storage.

An AIP contains zero or one *content information* part (an encapsulation of the raw data and its interpretation) and one or more *preservation description information* (PDI) parts (additional metadata related to the content logical preservation). The content information consists of the data we want to preserve, called the *content data object* (or *raw data*), and its associated *representation information* (RepInfo)—whatever additional information is necessary to ensure that the preserved data is intelligible to its designated community. For example, the RepInfo may consist of information about the hardware and software environment needed to view the content data object. The RepInfo may have additional RepInfo to interpret itself. The RepInfo recursion ends when it reaches a RepInfo that is nondigital and is considered to be preserved by the designated community. This chain of referenced RepInfo formed by the RepInfo recursion is called a *RepInfo network*. For example, astronomical data represented in a FITS (Flexible Image Transport System) file [27] has a RepInfo containing a dictionary that describes the FITS keywords. The FITS dictionary has a RepInfo containing the structure specification of the dictionary. Assuming the dictionary structure specification is in XML, its RepInfo

should include the XML specification, which in turn is associated with a RepInfo that includes the Unicode specification [28]. This is where the recursion ends; we assume that the Unicode specification is preserved by the designated community and, therefore, no additional RepInfo is required.

The PDI is divided into four sections, each describing the AIP content information. Additionally, the PDI has its own RepInfo to enable its interpretation over time. The PDI sections are as follows:

1. *Reference*—Consists of persistent identifiers out of which at least one is globally unique.
2. *Provenance*—Lists the chain of custody over the content information: its origin and the processing that it has gone through. For example, this chain of custody may include the time and date of each access to the preserved information.
3. *Context*—Documents the relationships between the content information and its environment. For example, if a chapter of a book is an AIP, then the context of the AIP of that chapter may include the references to the AIPs of the other chapters of the book.
4. *Fixity*—Proves that the content information was not altered in an undocumented manner.

PDS functionality

PDS 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. The PDS box is aware of the structure of an AIP and provides functionality for including representation information, tracking provenance, computing fixity, and supporting migration. Moreover, similar to the prototype system Abacus [29] and active disks [30], it can perform data-intensive functions within the storage. **Table 1** summarizes PDS functionality and the benefits it contributes to preservation environments. A more detailed discussion of these benefits can be found in [4].

PDS architecture

The PDS architecture (**Figure 1**) is based on the open standards OAIS, XAM, and OSD; each standard is implemented as a separate layer. At the top, the preservation engine layer, which is based on OAIS, provides an external interface to PDS and implements preservation functions. Additionally, it maps between the OAIS and XAM levels of abstraction, as detailed in [4]. XAM [25] serves as the storage mid-layer. It contains the XAM library, which provides the XAM interface, and a vendor interface module (VIM) to communicate with the underlying storage system. XAM provides a logical abstraction for data containers. It was chosen as the mid-layer abstraction because of its support for bundling extensive metadata with the data, its built-in support for import and export services, and its integrated storage management capabilities.

The bottom layer of the PDS architecture suggests two alternative back-end storage systems: a standard file system or an OSD. A higher-level application programming interface (API), labeled *HL OSD*, on top of the OSD provides abstraction and simplification to the object-based storage device (or *object store*) interface, which resembles a SCSI (small computer system interface).

Using a file system as the object layer is especially attractive when incorporating PDS with existing file systems and archives. However, there are advantages to the use of OSD, which is designed to support an object store layer [31]. Being able to manage space allocation and associate attributes with objects enables optimizations such as placing key OAIS attributes (e.g., a link to the RepInfos) close to the data in a persistent manner. Furthermore, in cases in which the actual disks are network attached, an OSD provides the networked disks with secure object-level access control. The scalability offered by OSD is also an important feature for preservation systems.

The PDS API can be triggered either directly or via Web services to enable flexible and platform-independent use of PDS. According to OAIS, the AIP is the basic object that interacts with the archival storage, and therefore, it is the main object to be passed between PDS and a client.

PDS is designed to run two processes: a high-level process and a low-level process. The high-level process is responsible for the PDS interface, the preservation engine, and the XAM implementation. The low-level process provides the object layer storage system, which is a file system or an object store. The two processes communicate via a remote procedure call (RPC) and are able to run on separate machines. The OSD, which is executed in the low-level process, requires occasional communication with third-party security administration; this is done via an API or Web service.

The PDS box incorporates the high-level process and possibly the low-level process. The architecture uses an application server to support HTTP (Hypertext Transfer Protocol), Web services, and the security interface with clients, as required for the PDS interface. This is in line with the new paradigm that advocates moving the application to the data instead of moving the data to the application [21].

PDS design and implementation issues

In this section, we describe several design and implementation issues that arose while developing the

Table 1 Preservation DataStores (PDS) functionality and benefits for preservation environments. (AIP: archival information package; PDI: preservation description information.)

Preservation-aware storage requirements	*PDS support*	*Benefits for preservation environments*
Encapsulate and physically colocate the raw data and its metadata objects, such as RepInfo, provenance, and fixity.	■ Awareness of the AIP structure. ■ Manage data availability at the level of an AIP. ■ Group related objects and create copies of objects according to the inherent importance of the data. ■ Aggregate the AIPs into clusters so that each cluster is self-contained and can be placed on the same media unit.	■ Ensures that metadata needed for interpretation is not separated from the raw data and thus reduces the risk of losing the metadata. ■ Supports graceful loss of data, namely the degree of lost information is proportional to the number of bits lost.
Execute data-intensive functions such as fixity computation within the storage.	■ Provide a storlets container (such as applets in applications and servlets in servers), a container that can embed and execute restricted modules with predefined interfaces.	■ Lessens network bandwidth. ■ Reduces risks of data loss. ■ Utilizes the locality property.
Execute transformations internally.	■ Provide a storlets container that can embed and execute transformations.	■ Simplifies applications since transformations can be carried out by the storage instead of the application. ■ Improves performance by reducing bandwidth. ■ Enables transformations to be applied during the migration process.
Include in the stored AIP the RepInfo of its PDI.	■ Awareness of the PDI structure.	■ Enables the interpretation of PDI in the future.
Handle technical provenance records internally.	■ Awareness of the provenance metadata and appending internally technical provenance events, such as events related to the migration.	■ Simplifies applications on top of PDS by reducing the number of events to handle. ■ Includes richer events that are known only to the storage.
Support media migration as opposed to system migration (i.e., migration by physically detaching the media from one system and attaching it to the new system).	■ Supporting a standardized self-contained, self-describing data format.	■ Reduces the cost of migration. ■ Reduces the risks of data loss during migrations.
Maintain referential integrity including updating all the links during the migration process so they remain valid in the new system.	■ Awareness of the context and RepInfo metadata and understanding the fields that represent links to either internal or external locations.	■ Simplifies applications. ■ Increases the robustness of the system.

PDS prototype. Our work so far provides some insights into the considerations and tradeoffs related to the various PDS layers. The implementation issues are ordered from the PDS bottom layer to the PDS upper layer.

HL-OSD API

The OSD was chosen to serve as back-end storage to the XAM layer in the PDS architecture. However, the current OSD initiator API supports extensive low-level SCSI-like functionality. In order to expose a more generic storage resource API rather than a storage device API, a higher-level API was defined on top of the current OSD initiator API. The goal was to provide a high-level, simple, and easy-to-use command set by adding wrappers and abstraction levels to existing OSD commands. In PDS, both OSD and its HL API are executed in a separate process from the upper layers. The XAM layer communicates with the HL-OSD interface via RPC.

XAM session execution model

A XAM storage system contains one or more *XSystems*, where each XSystem is a logical container of *XSet* records. An XSet, the basic artifact in XAM, is a data

structure that packages multiple pieces of XSet fields (data and metadata), bundled together for access under a common globally unique external name. A XAM client that requires access to a specific XSystem has to establish a XAM session with the PDS XAM library. A XAM session represents a path to the underlying object storage and serves as a context in which XAM requests can be performed. In order to perform a PDS API call (e.g., `ingestAIP`), multiple XAM requests are invoked on the same XAM session, for example, create an XSet and create the XSet various fields. Since the PDS API call is expected to be long-lived and to handle large amounts of data, multiple PDS API calls are not handled on the same thread. Instead, each such PDS API call is assigned with a dedicated XAM session executed in a dedicated thread.

Transactions in XAM

In order to support the XSet behavioral model, the concept of an *XSet transaction* was introduced. XSet transactions support atomicity, consistency, isolation, and durability. An XSet transaction may include one XAM API call (e.g., deleteXSet) or it may include several XAM API calls. For example, an XSet transaction may begin on openXSet, perform several calls, and end on closeXSet. If closeXSet is called without a prior commit call, the state of the XSet will be rolled back to the state before the transaction began. A commit call will make the changes persistent.

PDS interfaces

PDS exposes a set of interfaces that form the PDS entry points accompanied with their arguments and return values. The PDS entry points cover the functionality PDS exposes to its users including a variety of ways to ingest and access data and metadata, manipulate previously ingested data and metadata, retrieve PDS system information, and configure policies. The entry points may be called directly or via Web services. The PDS interfaces aim to be abstract and technology independent and to survive implementation replacements. The entry points may return different exceptions that are also PDS interfaces.

Some of the structures used by the PDS entry points, such as the inner structure of the PDI record (e.g., the inner structure of a provenance record), may have different variants and may depend on the source that generated the record. PDS aims to treat a set of records that may differ in their inner structure in a harmonic way: Although the provenance records may contain records with different inner structures, PDS still handles them similarly. To enable that, the inner structure of each record has by itself RepInfo, such as an XML schema, that is maintained along with the content of the record. When a record is generated by PDS, we use a PDS default XML schema as the RepInfo for the record. These PDS default schemas are also exposed so that users can make use of them.

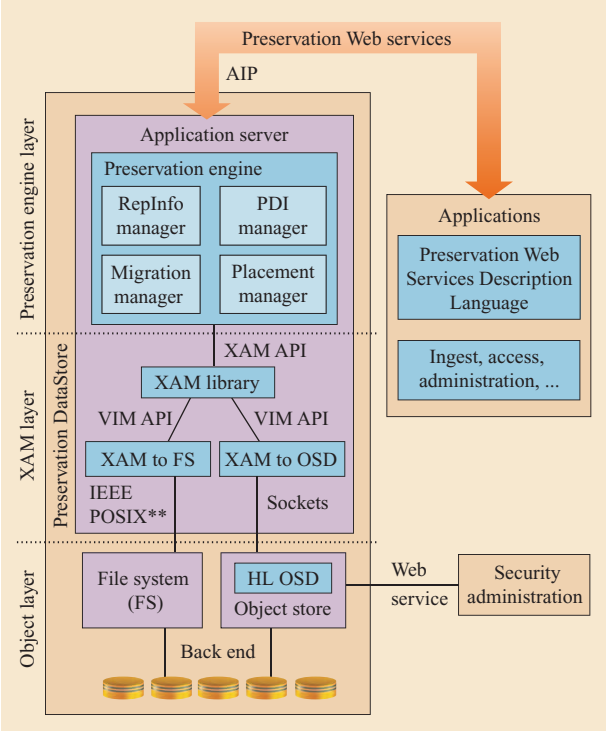

Figure 1

Preservation DataStores architecture. (AIP: archival information package; PDI: preservation description information; API: application programming interface; VIM: vendor interface module; OSD: Object-based Storage Device; HL: higher level.)

Because the implementation of each PDS layer as well as the whole PDS box will be replaced in the future with newer implementations, it is intended that the interface to PDS and the standards-based interfaces between the layers will survive longer. When these interfaces become obsolete, they will be migrated to newer interfaces developed in the future as technology advances. The preservation storage needs to be preserved by itself.

AIP identifier

Each AIP is assigned a persistent, globally unique identifier. In PDS, this identifier is constructed as follows: a *logical identification number* (ID) (which is the basic identifier of the AIP and is used to find the AIP in the preservation system), a *version identifier* (identifies the various versions originating from the same AIP), and a *copy identifier* (identifies the various copies of the same bit-wise AIPs).

As the PDS creates new AIPs, it needs to support AIP ID generation. In some cases, the newly created AIP may

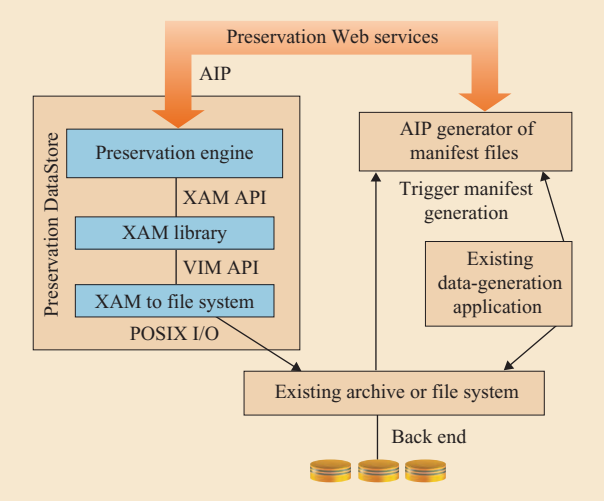

Figure 2

Preservation with existing file system.

be a new version or a new copy of an existing AIP in the system. In this case, the new AIP will have the same logical ID as the original AIP with a different version or copy identifier. In other cases, a brand new AIP may be created in the PDS; thus, it will have a new logical ID.

A new version AIP may be created when there is a change in the content information object (e.g., a transformation causes changes to the data and its RepInfo). External PDI events also cause the generation of a new version AIP (e.g., a provenance event that documents a change of AIP ownership). Note that the addition of technical PDI events (e.g., a provenance event that documents media migration) will update the PDI data but will not cause a new version AIP to be created. Completely new AIPs with new logical IDs may be created when preserving the RepInfo. For example, when ingesting an AIP with complex data and recursive RepInfo, PDS may generate several AIPs for the recursive RepInfo that reference each other. These AIPs are completely new and have different logical IDs.

Ingest AIP cost

To estimate the cost of ingesting a single AIP to the PDS in terms of space capacity, we need to consider two key variables: the number of AIPs created in the process and the size of each AIP. Since the ingest process of a single AIP may cause the creation of several AIPs, we need to estimate the actual number of AIPs created on ingest. This number is affected by the following two parameters:

1. *Complexity of the content data*—Complex data may be divided into several parts; each may have its own RepInfo. The ingest of such content will cause the creation of several AIPs aggregated by a main AIP.
2. *Number of RepInfo objects*—For each created AIP, there may be several RepInfo objects, each followed by more RepInfo objects that form the RepInfo networks. Each RepInfo object is represented by a separate AIP. Since RepInfo is a shared resource in the system, some of these RepInfo AIPs may already exist in the system. The ingest process will create only new AIPs for RepInfo objects that have not yet been introduced into the system. Clearly, as the number of AIPs in the system increases, the need to create new AIPs for RepInfo objects decreases, in particular when the AIPs represent content of similar formats.

The parameter that has the greatest effect on the size of a specific AIP is the size of its raw data. Therefore, the space cost of a single ingest may be estimated as the sum of raw data of the AIPs actually created in the process.

Integrating PDS with existing archives

In many cases, the data that is subject to long-term digital preservation already resides in existing archives. These archives may be simple file systems or more advanced archives that include enhanced functions such as metadata advanced query, hierarchical storage management, routine or special error checking, disaster recovery capabilities, and bit preservation. Some of this data is generated by applications that are unaware of the OAIS specification and the AIP logical structure and generally include just the raw content data with minimal metadata. While these archives are appropriate for short-term data retention, they cannot ensure long-term data interpretation at some arbitrary point in the future when everything—including hardware, software, processes, formats, and people—can become obsolete.

PDS can be integrated with existing file systems and archives to enhance such systems with support for OAIS-based long-term digital preservation. **Figure 2** depicts the architecture for such integration. We propose the addition of two components to the existing archive: an AIP generator of manifest files and a PDS box. The AIP generator generates AIPs, and each AIP is a manifest file with links to the existing content data and other metadata that already exist in the archive. If some metadata is missing (e.g., RepInfo), the AIP generator will be programmed to add that part either by embedding it into the manifest file or by saving it as a separate file or database entry linked from the manifest file. Sometimes, programming the AIP generator to generate those manifest files can be quite simple, for example, if there is an existing naming scheme that relates the various AIP parts.

The AIP generator can be triggered to generate a manifest file either by the existing archive or by the existing data-generation application. The former method is possible if the existing archive has the mechanism to trigger an event when new content data is ingested into the archive. The latter method is possible if the existing data-generation application has the mechanism to hook the AIP generator when new content data is generated. Note that in both cases data can be entered into the archive using the existing data-generation applications and will, thus, not require writing new applications.

The generated manifest AIPs are ingested into the second component: the PDS box. PDS provides most of its functionality—including awareness of the AIP structure and execution of data-intensive functions such as transformations—within the storage. It handles technical provenance records internally, supports media migration, and maintains referential integrity. However, PDS cannot guarantee intelligent data placement. It cannot guarantee physical colocation of the various parts that constitute the AIP, and it cannot guarantee the aggregation of related AIPs into clusters placed on the same media unit. However, when the next periodic migration phase happens because of media decay or for other reasons, PDS can optimize data placement and provide physical colocation of related AIPs at that time.

Conclusions

The problem of long-term digital preservation is becoming more real as we find ourselves in the midst of a digital era. Old assumptions regarding information preservation are no longer valid, and it is clear that aggressive actions are needed to ensure the understandability of data for decades to come. In order to address these changes, new technologies and systems are being developed. Such systems will be able to better address these vital issues if they are equipped with storage technology that is inherently dedicated to preservation and that supports the different aspects of the preservation environment. An appropriate storage system will make any solution more robust and lower the probability of data corruption or loss.

This paper describes PDS, an innovative storage architecture for OAIS-based preservation-aware storage. We described some of the design and implementation issues encountered while developing the PDS prototype. This prototype currently serves as an archival storage infrastructure component in the CASPAR project. We presented our architecture, which is composed of three layers of abstraction. The preservation layer, the compound object layer, and the stored-object layer are based on the OAIS, XAM, and OSD open standards, respectively. Each layer provides object abstraction using AIP, XSet, and OSD objects, linked by generic mappings.

Because data subject to long-term data preservation may already reside in existing file systems and archives, this paper describes the architecture for integrating PDS with those existing archives.

Acknowledgments

This work is partially supported by the European Community under the Information Society Technologies program of the sixth FP for RTD—project CASPAR contract IST-033572. The authors are solely responsible for the content of this paper. It does not represent the opinion of the European Community, and the European Community is not responsible for any use that might be made of data appearing therein.

References

1. *ISO Standard 14721:2003,* "Space Data and Information Transfer Systems—A Reference Model for an Open Archival Information System (OAIS)," International Organization for Standardization, 2003.
2. CASPAR—Cultural, Artistic and Scientific knowledge for Preservation, Access and Retrieval, CASPAR Project; see *http://www.casparpreserves.eu/*.
3. 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 Syst. Rev.* **41**, No. 1, 19–23 (2007).
4. M. Factor, D. Naor, S. Rabinovici-Cohen, L. Ramati, P. Reshef, J. Satran, and D. L Giaretta, "Preservation DataStores: Architecture for Preservation Aware Storage," *Proceedings of the IEEE Conference on Mass Storage Systems and Technologies*, San Diego, CA, 2007, pp. 3–15.
5. Y. Duhoux, "How Not to Decipher the Phaistos Disc: A Review" (review of J. Faucounau, *Le Déchiffrement du Disque de Phaistos: Preuves et Conséquences*), *Am. J. Archaeol.* **104**, No. 3, 597–600 (2000); see *http://www.ajaonline.org/archive/104.3/pdfs/AJA1043.pdf*.
6. M. Baker, K. Keeton, and S. Martin, "Why Traditional Storage Systems Don't Help Us Save Stuff Forever," *Technical Report HPL-2005-120*, HP Laboratories, 2005.
7. M. Baker, M. Shah, D. S. H. Rosenthal, M. Roussopoulos, P. Maniatis, TJ Giuli, and P. Bungale, "A Fresh Look at the Reliability of Long-Term Digital Storage," *Proceedings of the First ACM SIGOPS/EuroSys European Conference on Computer Systems*, Leuven, Belgium, 2006, pp. 221–234.
8. R. W. Moore and R. Marciano, "Building Preservation Environments," *Proceedings of the Fifth ACM/IEEE-CS Joint Conference on Digital Libraries*, Denver, CO, 2005, p. 424.
9. A. Rodriguez, "Preserving the Last Copy: Building a Long-Term Digital Archive," *Computer Technol. Rev.* **25**, No. 3, 17 (2004).
10. M. Peterson, "The Coming Archive Crisis," white paper, Storage Networking Industry Association (2006); see *http://www.snia.org/forums/dmf/programs/ltacsi/100_year/SNIA-DMF_The-Coming-Archive-Crisis_20061130.pdf*.
11. National Digital Information Infrastructure and Preservation Program (NDIIPP), The Library of Congress; see *http://www.digitalpreservation.gov/*.

12. M. W. Storer, K. Greenan, and E. L. Miller, "Long-Term Threats to Secure Archives," *Proceedings of the Second ACM Workshop on Storage Security and Survivability*, Alexandria, VA, 2006, pp. 9–16.
13. M. W. Storer, K. M. Greenan, E. L. Miller, and K. Voruganti, "POTSHARDS: Secure Long-Term Storage Without Encryption," *Proceedings of the USENIX Annual Technical Conference*, Santa Clara, CA, 2007, pp. 143–156.
14. R. W. Moore, J. F. JaJa, and R. Chadduck, "Mitigating Risk of Data Loss in Preservation Environments," *Proceedings of the 22nd IEEE/13th NASA Goddard Conference on Mass Storage Systems and Technologies*, Washington, DC, 2005, pp. 39–48.
15. The San Diego Supercomputer Center (SDSC) Storage Resource Broker (SRB); see *http://www.sdsc.edu/srb/index.php/Main_Page*.
16. iRODS: Intelligent Rule-Oriented Data System, San Diego Supercomputer Center (SDSC); see *http://irods.sdsc.edu/index.php/Main_Page*.
17. S. Quinlan and S. Dorward, "Venti: A New Approach to Archival Storage," *Proceedings of the Conference on File and Storage Technologies*, Monterey, CA, 2002, pp. 89–101.
18. L. L. You, K. T. Pollack, and D. D. E. Long, "Deep Store: An Archival Storage System Architecture," *Proceedings of the 21st International Conference on Data Engineering*, Tokyo, Japan, 2005, pp. 804–815.
19. E. Oltmans, R. J. van Diessen, and H. van Wijngaarden, "Preservation Functionality in a Digital Archive," *Proceedings of the 2004 Joint ACM/IEEE Conference on Digital Libraries*, Tucson, AZ, 2004, pp. 279–286.
20. P. Maniatis, M. Roussopoulos, TJ Giuli, D. S. H. Rosenthal, and M. Baker, "The LOCKSS Peer-to-Peer Digital Preservation System," *ACM Trans. Comp. Syst.* **23**, No. 1, 2–50 (2005).
21. K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. Seltzer, "Provenance-Aware Storage Systems," *Proceedings of the USENIX Annual Technical Conference*, Boston, MA, 2006; see *http://www.eecs.harvard.edu/syrah/pass/pubs/usenix06.pdf*.
22. PLANETS: Preservation and Long-Term Access through Networked Services; see *http://www.planets-project.eu/*.
23. The BRICKS Project: Building Resources for Integrated Cultural Knowledge Services; see *http://www.brickscommunity.org/*.
24. The DILIGENT Project: A Digital Library Infrastructure on Grid Enabled Technology; see *http://www.diligentproject.org/*.
25. C. Casten, "The Power of XAM," SNIA Data Management Forum; see *http://www.snia.org/about/resources/SNIA-DMF_The-Power-of-XAM_20060710.pdf/*.
26. *ANSI/INCITS 400-2004*, "SCSI Object-based Storage Device Commands (OSD)," InterNational Committee for Information Technology Standards (formerly INCITS), Project T10/1355-D, July 30, 2004; see *http://www.t10.org/ftp/t10/drafts/osd/osd-r10.pdf*.
27. FITS: Flexible Image Transport System, The NASA FITS Support Office; see *http://fits.gsfc.nasa.gov/*.
28. About the Unicode Standard, Unicode, Inc.; see *http://www.unicode.org/standard/standard.html*.
29. K. Amiri, D. Petrou, G. R. Ganger, and G. A. Gibson, "Dynamic Function Placement for Data-Intensive Cluster Computing," *Proceedings of the USENIX Annual Technical Conference*, San Diego, CA, 2000, pp. 307–322.
30. E. Riedel, C. Faloutsos, G. A. Gibson, and D. Nagle, "Active Disks for Large-Scale Data Processing," *IEEE Computer* **34**, No. 6, 68–74 (2001).
31. M. Factor, K. Meth, D. Naor, O. Rodeh, and J. Satran, "Object Storage: The Future Building Block for Storage Systems—A Position Paper," *Proceedings of Local to Global Data Interoperability—Challenges and Technologies*, Sardinia, Italy, 2005, pp. 119–123.

Received October 1, 2007; accepted for publication February 13, 2008; Internet publication June 3, 2008

Simona Rabinovici-Cohen *IBM Haifa Research Laboratory, Haifa University Campus, Mount Carmel, Haifa 31905, Israel (simona@il.ibm.com)*. Ms. Rabinovici-Cohen holds B.S. and M.S. degrees in computer science, both from the Technion, Israel Institute of Technology. Since 1993, she has been a Research Staff Member at the Haifa Research Laboratory. She worked on various projects in the area of information integration and knowledge management systems, including data integration for performance management, integrated medical records, de-identification, and biomedical information integration. Ms. Rabinovici-Cohen was a technical leader of the IBM clinical-genomics solution. Currently, she works on technology to support long-term digital preservation.

Michael E. Factor *IBM Haifa Research Laboratory, Haifa University Campus, Mount Carmel, Haifa 31905, Israel (factor@il.ibm.com)*. Dr. Factor holds a B.S. degree in computer science from Union College, and M.S., M.Phil., and Ph.D. degrees in computer science, all from Yale University. He is an IBM Distinguished Engineer with a focus on storage and systems. He has been an architect of advanced copy functions for the IBM DS family of storage subsystems and takes a leading role in storage-related research, including such topics as long-term digital preservation, storage power, and object-based storage. Dr. Factor is chair of the security subgroup of the SNIA OSD standardization effort. In the past, he was the manager of Distributed and Clustered Systems at the Haifa Research Laboratory. Previous projects he was involved with include the Cluster Virtual Machine for Java**, the XML file system, the IBM iSeries* integrated file system, and the Web server for the 1996 Atlanta Olympics.

Dalit Naor *IBM Haifa Research Laboratory, Haifa University Campus, Mount Carmel, Haifa 31905, Israel (dalit@il.ibm.com)*. Dr. Naor holds M.S. and Ph.D. degrees from the University of California, Davis, and a B.S. degree from the Technion, Israel Institute of Technology, all in computer science. She is a Research Staff Member and a manager of the Storage Systems and Performance Management group. She is developing advanced functions for storage systems, including support for long-term digital preservation and secure access to storage. She contributed to the development of the object storage T10 OSD standard, a new object-based storage interface. Dr. Naor's prior areas of interests include combinatorial optimization, bioinformatics, applied security, and content protection.

Leeat Ramati *IBM Haifa Research Laboratory, Haifa University Campus, Mount Carmel, Haifa 31905, Israel (leeat@il.ibm.com)*. Ms. Ramati is a Research Staff Member in the Storage Systems and Performance Management group. She received her B.Sc. in computer science *summa cum laude* from the Technion, Israel Institute of Technology. Since joining IBM, Ms. Ramati has worked on technology to support long-term digital preservation, and she is a XAM expert.

Petra Reshef *IBM Haifa Research Laboratory, Haifa University Campus, Mount Carmel, Haifa 31905, Israel (petra@il.ibm.com)*. Mrs. Reshef is a Research Staff Member in the Storage Systems and Performance Management group. She holds a B.S. degree in computer science from the University of Haifa. Her work has focused on storage and network technologies: SCSI, Internet SCSI (iSCSI), Fibre Channel Protocol, and OSD. Since 2006, she has been working on technology to support long-term digital preservation, developing a prototype for the storage component within a preservation system.

Shahar Ronen *IBM Haifa Research Laboratory, Haifa University Campus, Mount Carmel, Haifa 31905, Israel (shaharr@il.ibm.com)*. Mr. Ronen is a Research Staff Member in the Storage Systems and Performance Management group. He holds a B.S. degree in computer science from the University of Haifa. Since joining IBM in 2006, he has worked on technology to support long-term digital preservation.

Julian Satran *IBM Haifa Research Laboratory, Haifa University Campus, Mount Carmel, Haifa 31905, Israel (julian_satran@il.ibm.com)*. Mr. Satran holds an M.S.E.E. degree from the Polytechnic Institute of Bucharest, Romania. He is an IBM Distinguished Engineer. His current areas of interest span system and subsystem architecture, networking, and development and operating environments. He led several pioneering research projects in clustering, file system structure (and object storage), I/O and networking convergence (iSCSI), and future, more rational and scalable I/O subsystems. He has driven an industry-wide effort to standardize iSCSI and is now leading an effort to standardize object storage. He is a member of IEEE and ACM.

David L. Giaretta *Science and Technology Facilities Council, Rutherford Appleton Laboratory, Harwell Science and Innovation Campus, Didcot, Oxon OX11 0QX, United Kingdom (d.l.giaretta@rl.ac.uk)*. Dr. Giaretta is the Project Director of two European Union co-funded projects, CASPAR and PARSE.Insight, as well as the Associate Director for development in the Digital Curation Centre. He holds an M.Sc. degree in mathematics, an M.A. degree in physics, and a D.Phil. degree in theoretical physics and astrophysics, all from Oxford. He was awarded an MBE (Member of the British Empire) in 2003 for services to space science. Dr. Giaretta has had extensive experience in planning, developing, and running scientific archives and developing scientific applications. He chaired the panel that produced the Open Archival Information System (OAIS) Reference Model (ISO 14721) that forms the basis of much digital preservation work and is contributing to the development of many of the follow-on standards.