Simona Cohen, Michael Factor, Dalit Naor

Preservation DataStore (PDS): a demonstration of Preservation Aware Storage

slide 1 Preservation DataStores, or PDS, is a preservation-aware open archival storage system being developed at the IBM Haifa Research Lab.

slide 2 PDS is preservation-aware storage, which means that the storage component of a digital preservation system has built in support for both bit preservation and logical preservation.

The storage is aware of the OAIS based preservation objects that it stores and is capable of offloading functionality that is traditionally performed by the application. Among these functions you can find: handling metadata, calculating and validating fixity, documenting provenance events, managing the Representation Information (RepInfo) of the Preservation Description Information (PDI), and validating referential integrity.

PDS supports loading and execution of storlets, which are execution modules for performing data intensive functions such as data transformations and fixity calculations close to the data.

Offloading metadata management and data intensive functions to storage decreases the probability of data loss by minimizing data transfers between storage and the application. It simplifies applications by decreasing the need for data management at the application level and provides improved performance and robustness.

Another important feature is the physical co-location of data and metadata, which ensures that metadata is not lost if raw data survives. Related Archival Information Packages (AIPs) are also co-located on the same media.

PDS is able to generate AIP identifiers to satisfy the cases in which AIPs are created within PDS or cases in which AIPs are ingested without an assigned AIP ID.

slide 3 PDS is built on layers, each based on open standards.

The preservation engine layer implements the PDS API, which is externalized using web services. It's based on OAIS.

The XAM mid-layer is based on the XAM emerging standard that provides complex objects for bundling pieces of data and metadata and is specialized for reference data. The object layer is currently based on the OSD standard.

slide 4 Here you can see how an AIP is represented in the preservation engine.

An AIP is built from information objects. Each information object contains a single instance of data, and zero or more instances of RepInfo. An AIP contains the content information, which is an information object.

Each section of the PDI is also an information object. For instance, the provenance contains the provenance data that lists the events in the life of the data. Its RepInfo may describe the specific structure in which the provenance data is kept.
In PDS, RepInfo is also an AIP, since RepInfo is a shared resource in the system that requires a unique ID. It has its own RepInfo, thus forming the RepInfo recursive network.

slide 5

The preservation process of an AIP can be distinguished into 3 phases: the ingest process in which the AIP is stored, the access process where the AIP is retrieved, and the preservation process between them.
While the data is preserved we need to maintain the ability to restore the bits and to provide bit preservation. Bit migrations transfer the bits from one media to another when the media becomes obsolete.
To maintain the understandability of the data. we need to keep the data and accompanying metadata updated. This means performing format migrations when the format becomes obsolete and continuing to update the PDI and RepInfo.
For example, a change in the knowledge base of the designated community may require adding RepInfo.
Fixity calculations need to be performed periodically and provenance updates need to occur at each event in the life of the data such as migration.
When we conduct data transformations, the transformation module is packaged as an AIP and preserved. The transformation result is also preserved as a separate AIP, along with the original AIP, representing another version of the original. Its AIP ID will be a version of the original data AIP ID to maintain the connection between different versions of the same data.
As a result, by retrieval time, the original AIP may have several versions and copies.

slide 6

The following was presented in the CASPAR 2nd review and involved the Registry, Packaging and PDS components. It used MST atmospheric data from the CASPAR scientific testbeds.
The scenario showed how the atmospheric scientists transited data from one format to another, and how this transit affected the preserved data, keeping its usability and understandability in light of this change.
The following demonstration shows the PDS archival storage functionality that was used in that demo. You will be able to see how the PDS GUI is used manually to perform these functions, while complying with the OAIS preservation model.

slide 7

To start with, here is some background on the data.
MST data provides wind measurements, measured by a specific radar placed in the UK. This data is highly complex and preserved for the atmospheric scientific community.

It requires preservation for a long period of time for comparative research.
It should be kept in a format that is supported by the majority of the community to enable interoperability.

slide 8 The British Atmospheric Data Center recommends using a self-describing format for this complex data.
Currently 2 formats compete in the community: NASA-AMES, which is an ASCII format, and NetCDF, which is a binary format.
The current decision is to use the NetCDF format.

slide 9 In the following demo you will see how this data is ingested and preserved in PDS, and how PDS supports its continuous understandability and usability.
Let's see how this AIP works.

slide 10 The MST AIP is packaged in XFDU packaging format. It contains a manifest file that organizes the package and contains some embedded metadata.
The raw data and some of the metadata are provided in files external to the manifest. All these files are packed in a single ZIP file.
If you take a look inside the data file you can see the NetCDF self-describing header and the binary data.

slide 11 To preserve this AIP, we ingest it to PDS.
Inside PDS, on ingest, the ingest web service is called with the packaged AIP.
The preservation engine unpacks the AIP, generates an AIP ID for it, computes fixity, and adds an "ingest" provenance event.
It then creates XAM objects and maps the AIP into them.
These XAM objects are then mapped to OSD objects and stored at the OSD.
On the way back, the AIP ID is returned to the caller.

slide 12 Next you will see how PDS GUI is used to ingest the MST AIP.

slide 13 And here is the PDS GUI that we use to trigger the PDS functions published as web services by the PDS server.
We choose the ingestAip method, browse for the MST AIP, and send the request to the PDS server.
The return value is the AIP ID assigned by PDS.
After ingest, the different sections of the AIP can be accessed separately. For example, the content data RepInfo.
We choose the accessRepInfo method, insert the AIP ID we received on ingest, and send the request.
The returned table includes the RepInfo records of the MST NetCDF AIP. The first RepInfo record refers to RepInfo that was stored in the Registry and the CPID is provided.
The other RepInfo records refer to RepInfos that were embedded in the AIP and therefore PDS ingested each of them as a separate AIP. The resulting AIP IDs

were stored as RepInfo records.

slide 14 Let's suppose some time passes.

slide 15 For several reasons it is decided to perform format migration of the preserved MST data from NetCDF to NASA-AMES format.
Perhaps the NetCDF format is no longer supported, the community became biased towards ASCII formats, and NASA-AMES has become the preferred data format of the atmospheric scientists community.
In addition, it was decided to add a second view path to the data, providing an image view for users with limited access permissions.

slide 16 To perform a transformation, you need to load a transformation module to PDS.
Inside PDS, on LoadTransformation, an AIP that contains a transformation module is loaded into PDS.
This load operation includes the registration of this transformation module in the system and a regular ingest of its AIP into the archival storage system.

slide 17 After the transformation module is loaded, we trigger the transformation by calling the TransformAip method.
Inside PDS, on TransformAip, both the target AIP and the transformation AIP are accessed. The transformation is applied to the content data of the target AIP.
To preserve the result, PDS builds a new AIP that contains the transformed data, new RepInfo to interpret it (which is supplied by the transformation AIP), and an appropriate PDI that is generated internally by PDS.
The new AIP is a version of the original AIP.

slide 18 Next, you will see how the PDS interface is used to maintain the usability and understandability and to support the interoperability of the preserved data.
A transformation module that converts MST NetCDF data to NASA-AMES format is loaded and executed in PDS.

slide 19 We choose the LoadTransformation method, and similar to ingest, browse for the transformation AIP that should transform NetCDF to NASA-AMES. We then send the request.
The return value is the AIPID assigned by PDS.
We choose the TransformAip method, and supply the AIP ID of the MST NetCDF AIP and the AIP ID of the transformation AIP from NetCDF to NASA-AMES.
The return value is the AIPID assigned by PDS to the new AIP that contains the MST data in NASA-AMES format.
This AIPID is a version of the MST NetCDF AIP. They share the logical ID part and have different version IDs.
In the same manner, we perform a second transformation from NetCDF to PNG

plot, which results in another version of the original AIP.

slide 20 Suppose more time passes.

slide 21 Inside PDS, on access, the Preservation Engine looks up the XAM ID that corresponds with the received AIP ID, and reads the appropriate XAM objects.
XAM retrieves its objects from the object layer.
When the AIP is in the Preservation Engine, it validates the AIP ID and its fixity, and adds a provenance event to document this access operation.

slide 22 This brings us to the last phase of this demo, which shows how PDS preserves the MST data.

slide 23 First, we query PDS for all the versions originating from the same MST NetCDF AIP.
We choose the QueryAipIdSet method and supply the MST NetCDF AIPID.
The result is 3 versions for that AIP: NetCDF format, NASA-AMES format, and PNG plot.
Now we access each of these AIPs.
PDS gives us separate access to the different AIP sections.
First, we access the content data of the MST NASA-AMES AIP.
We choose the accessContentDataAsLink method that places the content data in a staging area for downloading.
We supply the MST NASA-AMES AIPID and send the request.
The return value is a reference to a staging area for downloading the content data.
You can see the MST data in NASA-AMES ASCII format.
Since PDS created a complete AIP for this data transformation result, we can also access the PDI metadata sections. First, we access the Provenance...
We choose the accessProvenance method and supply the MST NASA-AMES AIPID.
The result is a table that contains the provenance records of the MST NASA-AMES AIP. The first record describes the creation of the AIP, which was the result of data transformation. The others document the ingest and access operations.
Then, we access the Fixity.
We choose the accessFixity method and supply the MST NASA-AMES AIPID.
The result is the fixity record generated by PDS, which documents the fixity calculation that took place when this AIP was ingested.
Then we access the Reference.
We choose the accessReference method and supply the MST NASA-AMES AIPID.
The result includes a single reference record that was generated by PDS and contains the AIP ID of this AIP.
Finally, we access the content data of the MST PNG plot AIP.

Again, we choose the access Content Data As Link method, and supply the MST PNG AIP ID.
Now you can see the MST data as a plot in PNG format. This provides us with the requested additional view path.

slide 24

In this demonstration, you saw how PDS can be used to ingest an AIP and access its different sections.
PDS supports the continuous usability and understandability of the ingested data by enabling us to load and execute transformation modules and by maintaining the metadata sections of the AIP.