Abstract

"Collection-Based Persistent Digital Archives: Part 2" describes the creation of a one million message persistent E-mail collection. It discusses the four major components of a persistent archive system: support for ingestion, archival storage, information discovery, and presentation of the collection. The technology to support each of these processes is still rapidly evolving, and opportunities for further research are identified.

1. Collection Support, General Requirements

Persistent archives can be characterized by two phases, the archiving of the collection, and the retrieval or instantiation of the collection onto new technology. The processes used to ingest a collection, transform it into an infrastructure independent form, and store the collection in an archive comprise the persistent storage steps of a persistent archive. The processes used to recreate the collection on new technology, optimize the database, and recreate the user interface comprise the retrieval steps of a persistent archive. The two phases form a cycle that can be used for migrating data collections onto new infrastructure as technology evolves. The technology changes can occur at the system-level where archive, file, compute and database software evolves, or at the information model level where formats, programming languages and practices change.

1.1 Collection Process Definition

The initial data set ingestion and collection creation can be seen as a process in which:
(a) objects are captured, wrapped as XML digital objects, and categorized in a (relational) database system,

(b) the collection is ingested into an archival-storage system using containers to hold the digital objects along with all pertinent meta-data and software modules.

The migration cycle can be seen as the reverse of the ingestion process in which:

(a) containers are brought out of deep-store and loaded into a (possibly NEW) database system (that can be relational or hierarchical or object oriented),

(b) the database is queried to form (possibly NEW) containers that are placed back into a (possibly NEW) archival storage system.

Note the similarities in steps (a) and (b) of the ingestion and migration processes. In order to build a persistent collection, we consider a solution that "abstracts" all aspects of the data and its preservation. In this approach, data objects and processes are codified by raising them above the machine/software dependent forms to an abstract format that can be used to recreate the object and the processes in any new desirable forms.

Data objects are abstracted by marking the contents of each digital object with tags that define the digital object structure. Tags are also used to mark the attributes that are used to organize the collection and define the collection context. Processes are abstracted such that one can create a new "procedure" in a language of choice. Examples are the ingestion procedures themselves. They comprise "abstract load modules" for building the collection. Similarly, the querying procedures can be represented as "abstract mappings from a definition language to a query language" and visualization presentation procedures can be cast as "style-sheet abstractions".

The multiple migration steps can be broadly classified into a definition phase and a loading phase. The definition phase is infrastructure independent, whereas the loading phase is geared towards materializing the processes needed for migrating the objects onto new technology.

We illustrate these steps by providing a detailed description of the actual process used to ingest and load a million-record E-mail collection at the San Diego Supercomputer Center (SDSC). Note that the SDSC processes were written to use the available object-relational databases for organizing the meta-data. In the future, it may be possible to go directly to XML-based databases.

I. Ingestion/Storage Phase

The SDSC infrastructure uses object-relational databases to organize information. This makes data ingestion more complex by requiring the mapping of the XML DTD semi-structured representation onto a relational schema. Two aspects of the abstraction of objects need to be captured: relationships that exist in and among the data, and hierarchical structures that exist in the data. These were captured in two different types of abstractions: through a relational Data Definition Language (DDL), and through a semi-structured Document Type Definition. The relational abstraction is a mature technology that facilitates querying about the meta-data, whereas the semi-structured
abstraction is our chosen uniform information model. Hence, our process used both of these technologies to manage digital objects. In the future, with the emergence of XML-based database systems, only the semi-structured representation will be needed. In the model below, only the XML-DTD was stored as part of the abstract object; instead of storing the DDL, we stored the procedure for creating a DDL from a DTD. A system-dependent DDL was created using the DTD and the DTD-to-DDL mapping procedure with the addition of system-specific information. The software that creates the system-dependent DDL comprises the instantiation program between the digital objects stored in the archive, and the collection that is being assembled on new technology.

The steps used to store the persistent archive were:

Define Digital Object

- define meta-data
- define object structure (OBJ-DTD) --- (A)
- define object DTD to object DDL mapping --- (B)

Define Collection

- define meta-data
- define collection structure (COLL-DTD) --- (C)
- define collection DTD structure to collection DDL mapping --- (D)

Define Containers

- define packing format for encapsulating data and meta-data (examples are the AIP standard, Hierarchical Data Format, Document Type Definition)

II. Load Phase

The load phase uses the information models that were archived in the ingestion phase. The information models are read out of the archive and used to drive the database instantiation software:

Create generator for the Database-DDL --- (E)

- [(A),(B),(C),(D),Target-system Info] ==> COLL-DDL --- (X)

Create generator for the database Loader --- (F)

- [(A),(C),(X),Target-system Info] ==> DB Load-module --- (Y)

Create generators for presentation interface and storage --- (G)

- [(A),(C),(X),Target-system Info] ==> SQL & Style-Sheet --- (Z)
- [(A),(C),(X),Target-system Info] ==> Archive Load-module --- (Z')
Generate Containers and Store

- Store also (A),(B),(C),(D),(E),(F),(G) as part of packed format.

In the ingestion phase, the relational and semi-structured organization of the meta-data is defined. No database is actually created, only the mapping between the relational organization and the object DTD. Note that the collection relational organization does not have to encompass all of the attributes that are associated with a digital object. Separate information models are used to describe the objects and the collections. It is possible to take the same set of digital objects and form a new collection with a new relational organization.

In the load phase, the mappings between the relational and semi-structured representations of the objects and collections are combined with the semi-structured information model to generate the relational representation for the new database on the target system. The information is encapsulated as a software script that can be used to create the new database tables for organizing the attributes.

A second script is created that is used to parse the digital objects that are retrieved from the archive, and load the associated meta-data into the new database tables.

Steps (A),(B),(C) and (D) can be interpreted as abstract mark-up formats for digital objects and steps (E),(F) and (G) can be interpreted as abstract procedures. The formats and procedures can be combined to support migration of the collection onto new software and hardware systems, as well as migration onto new information models or data formats and new procedure languages.

In a system-level migration process, a database is created using (X), the database is instantiated from the copy of the container(s) in the archival storage system using (Y), and the data in the database is stored into a new archival storage system using (Z').

In a format-level migration process, new versions of (A),(B),(C),(D) are created based on the prior values, a database is created using the original (X) and instantiated with the prior (Y), and then the data in the database is reformatted and stored in the archival storage system using the new (Z').

In a language-level migration process, new versions of (E),(F),(G) are created based on the original values, and stored as part of the packaged container. The data itself is not migrated.

2. Persistent Archive Demonstration

The steps required to ingest a collection, archive the data and the collection description, and then re-create a database and support queries against the collection have been demonstrated as part of a persistent archive prototype. A collection of 1 million News group (Usenet) messages was created from technical topic groups, including Computer Science, Science, Humanities, and Social Science. The Usenet collection was chosen because RFC 1036 provides a standard for Usenet messages that defines both required and optional attributes.
The 1-million record E-mail collection was ingested, archived, and dynamically rebuilt within a single day. This was possible because all steps of the process were automated. The demonstration is scalable such that archiving of 40-million E-mail records can be done within a month. The steps for the 1-million record demonstration included assembling the collection, tagging each message using XML, archival storing of the digital objects, instantiating a new collection, indexing the collection, presenting the collection through a Web interface, and supporting queries against the collection. The required Usenet meta-data attributes form a core set that can be applied to all E-mail messages.

2.1 Ingestion Process

A typical raw E-mail record contains keywords that represent either required attributes, such as Path, From, Newsgroups, Subject, Date, Message-ID, optional keywords such as Organization, Mime-Version, Content-Type, and other user defined keywords that follow a defined syntax. The keywords can be used to identify a specific message, and therefore constitute the attributes used to organize the E-mail collection.

A DTD was derived which reflects the RFC1036 structure. Each of the required, optional, and other keyword items were associated with a seqno attribute used to record information on the sequence in which the various keywords appear in the original document. The order of appearance of keywords may be different in different documents. In the ingestion step, each E-mail record was parsed into the standard DTD. As a result, the messages can be displayed using Microsoft Notepad, an XML viewer, by applying the DTD. This provides the ability to impose a presentation style on the objects in a collection.

Loading sources with regular structure into a relational database (RDB) has several benefits:

- Inconsistencies in the data can be automatically detected using the RDB’s built-in consistency checks (data types, uniqueness of keys, referential integrity, etc.),

- Powerful ad-hoc SQL queries can be used to further clean the data from inconsistencies,

- Interesting information from the collection can be mined,

- Different versions of the collection can be compared,

- Using an RDB-to-XML wrapper provides an XML view on the collection.

We created 3 hand-crafted tables for the Newsgroup collection for the relational database implementation:

- The first table contained all the required and optional header field information supported in RFC1036. Additionally, an internalMsgId was used for cross-references with other tables, and sequence numbers were used to show the sequence in which the fields appeared in the original message.
• The second table contained facilities for storing other header fields not supported by RFC1036.

• The third table contained "systemic" information about how the body of messages were stored. A dataid field was used to define a "file" or "container" id. A posInContainer field was used to define the offset of the start of that record's body text, and a sizeOfMsg field was used to define the length in bytes of that record's body text.

Note that the semi-structured nature of the E-mail messages is more easily represented with an XML semi-structured representation.

To store the collection, the E-mail messages were aggregated into 25 containers, each holding about 40,000 records. Note that during ingestion, unique tags were added to define the beginning and end of each message. This is required because it is possible for an E-mail message to contain an encapsulated E-mail message, making it difficult to create digital objects by explicit analysis of the complete collection. In the current implementation of the data handling system, a "container" file can be registered with SRB/MCAT (the SDSC data handling system). SRB/MCAT then provides access mechanisms for retrieving individual messages from such containers.

The ingestion process was carried out on a workstation. The system was an SGI Indigo 2 (MIPS R10000 Processor, 195 Mhz, Memory size: 128 Megabytes). The collection size was 2.6 Gbytes. It took 12 hours to assemble the collection from the Newsgroup storage, 1 hour 39 minutes to parse the raw collection into an XML DTD, and 1 hour to store the containers into HPSS.

2.2 Instantiation Process

The instantiation process included the retrieval of the data sets from the HPSS archive, the creation of a load file for insertion into an Oracle database, the optimization of the Oracle index, and support for a query against the collection. The index optimization step decreased the time needed to do a query against the collection from 20 minutes to 1 second.

The time needed to create a database holding the collection included 1 hour to retrieve the data from the archive, 2 hours 40 minutes to build the load file, 4 hours to load the database, and 4 hours to optimize the database index.

3. Remaining Technical Issues

The four major components of the persistent archive system are support for ingestion, archival storage, information discovery, and presentation of the collection. The first two components focus on the ingestion of data into collections. The last two focus on access to the resulting collections. The technology to support each of these processes is still rapidly evolving. Hence consensus on standards has not been reached for many of the infrastructure components. At the same time, many of the components are active areas of research. To reach consensus on a feasible collection-based persistent archive, continued research and development is needed. Examples of the many related issues are listed below:
Ingestion

- Creation of a standard digital representation of the original (or raw) data. What unique tags should be used to define digital objects within the original raw data?

- Techniques for automating the decomposition of a data collection into individual digital objects. How can digital objects be defined when they must be extracted from proprietary formats?

- Automation of the mining of attributes used to describe each data object. Can a generic technique be developed that works for a class of data such as E-mail, or word processing documents?

- Standard information model for characterization of the data collection organization. This will require defining standard semantics as well as a standard for describing the collection structure.

- Representation of unique procedures associated with each collection, including software access tools and ingestion update tools. Can these tools be made interoperable across multiple collections, or will unique tools be required for each collection?

- Standardization of the mark-up language used to annotate the digital objects with their associated meta-data. Extensions are being proposed to XML to associate semantics with the tags, and define required structures within the DTD.

- Support for security within the ingestion process. What risks are incurred by use of common infrastructure for ingesting data at different security levels?

- Validation of ingestion process. Policies for validating the correctness of an infrastructure independent representation of a digital object are needed. Our XML approach did not capture white space.

- Workflow management policies. Workflow management policies are needed as a component of the ingestion process, to ensure that all validation steps are completed. Can validation be done after the fact through analysis of the collection, or should the validation be confirmed as the digital objects are created?

- Evolution of information models. There is a need for finding aids that are robust under evolution, and capable of locating all data collections stored within the persistent archive.

- Collection access. Access mechanisms are needed that are capable of handling changes to collections, such as construction of new indexes of collections, updating of collections by addition of objects, updating of collections by addition of new attributes, and updating through evolution of the DTD.

- Performance optimization for incremental updates of collections, and incremental updates of DTDs.
• Administrative tools for managing collections, including compaction of collections, updates to collections, and restructuring of collections.

• Derivation of DTDs to describe complex, semi-structured and unstructured collections.

• Support for heterogeneous collections, especially multi-media, graphical and web-based collections.

Archival storage

• Standardization of the archive format for storing a digital object based upon OAIS.

• Standardization of container formats for aggregating digital objects.

• Choice of digital objects to aggregate within containers for retrieval optimization from the archive.

• Standardization for registration of the collection within a finding aid to guarantee the ability to retrieve the data collection from the archive.

• Versioning of the information model to track changes.

• Support for migration of data between time dependent security levels.

Information discovery

• Development of generic software that is able to parse DTDs and generate appropriate commands for creating a new database.

• Support for dynamic reconstruction of the data collection through use of XML-based database technology. Additional attributes may need to be defined to manage evolution of the semi-structured representation.

• Support for dynamic generation of a user interface to support queries against data in XML databases.

• Dynamic generation of the query language required to access XML databases.

• Support for creation of queries against collections whose schema has evolved.

• Access control mechanisms and standards for managing classified information.

Presentation

• Standardization of the mark-up language used to define the presentation layout (XSL style sheets). An example is optimization of the layout of the display for user efficiency and ease of use.
Support for retrieval and presentation of meta-data used to characterize information about the object or the associated data collection. Can standard DTDs be used to organize meta-data for presentation?

Dynamic creation of the presentation interface for each digital object. Presentation cannot be a function of only the collection. For heterogeneous collections, the style of presentation must be defined for each type of object within the collection.

3.1 Research Opportunities

Important research areas that we suggest pursuing include:

- **Security**: So far we have dealt with unencumbered data ingestion. What happens when particular data elements need to reside at certain locations or when notions of data access control come into play?

- **Federation**: Can a persistent archive be distributed? DTD manipulation during distributed ingestion and integration of multiple DTDs are associated topics.

- **Workflow**: The validation process requires the guaranteed execution of analysis routines. Workflow management tools are needed to ensure that no processing steps are missed.

- **Complex collections**: What is the correct information model when dealing with multimedia data and GIS collections?

3.2 Summary

Multiple communities across academia, the federal government, and standards groups are exploring strategies for managing very large archives. The persistent archive community needs to maintain interactions with these communities to track development of new strategies for data management and storage. The technology proposed by SDSC for implementing persistent archives builds upon interactions with many of these groups. Explicit interactions include collaborations with Federal planning groups [1], the Computational Grid [2], the digital library community [3], and individual federal agencies [4].

The proposed persistent archive infrastructure combines elements from supercomputer centers, digital libraries, and distributed computing environments. The synergy that is achieved can be traced to identification of the unique capabilities that each environment provides, and the construction of interoperability mechanisms for integrating the environments. The result is a system that allows the upgrade of the individual components, with the ability to scale the capabilities of the system by adding resources. By differentiating between the storage of the information content and the storage of the bits that comprise the digital objects, it is possible to create an infrastructure independent representation for data organized by collections. Collection-based persistent archives are now feasible that can manage the massive amounts of information that confront government agencies.

Acknowledgements:
The data management technology has been developed through multiple federally sponsored projects, including the DARPA project F19628-95-C-0194 "Massive Data Analysis Systems," the DARPA/USPTO project F19628-96-C-0020 "Distributed Object Computation Testbed," the Data Intensive Computing thrust area of the NSF project ASC 96-19020 "National Partnership for Advanced Computational Infrastructure," the NASA Information Power Grid project, and the DOE ASCI/ASAP project "Data Visualization Corridor." Additional projects related to the NSF Digital Library Initiative Phase II and the California Digital Library at the University of California will also support the development of information management technology. This work was supported by a NARA extension to the DARPA/USPTO Distributed Object Computation Testbed, project F19628-96-C-0020.

References


Copyright © Reagan Moore, Chaitan Baru, Arcot Rajasekar, Bertram Ludaescher, Richard Marciano, Michael Wan, Wayne Schroeder, and Amarnath Gupta

DOI: 10.1045/april2000-moore-pt2