Collection-Based Persistent Archives

Arcot Rajasekar, Richard Marciano, Reagan Moore
San Diego Supercomputer Center

Abstract

The supercomputer center, digital library, and archival storage communities have common persistent archival storage requirements. Each of these communities is building software infrastructure to organize and store large collections of data. An emerging common requirement is the ability to maintain data collections for long periods of time. The challenge is to maintain the ability to discover, access, and display digital objects that are stored within the archive, while the technology used to manage the archive evolves. We propose an approach based upon the storage of the digital objects that comprise the collection, augmented with the meta-data attributes needed to dynamically re-create the data collection. This approach builds upon the technology needed to support extensible database schema, which in turn enables the creation of data-handling systems that support replicated data sets within federated archives.

I. Introduction

Archival storage systems provide support for the long-term storage of digital objects [1,2]. Each object is typically owned and managed by a researcher who individually deposits the data set into the archive using Unix pathname semantics. The system works well for storing data sets, but provides no support for managing the information needed to interpret or discover archived data sets. Current trends in data and information generation are leading to a paradigm shift in storing and manipulating data sets. This is driven by the creation of large collections of data (digital sky surveys will access over 2 billion images) and collections of large data objects (a brain image in a neuroscience database will eventually be as large as a terabyte) [3,4]. For scientific disciplines to survive under the onslaught of massive data loads, an efficient infrastructure needs to be developed to provide automated means of information ingestion, management, querying and access by future computations. Hence, one needs to consider a system that not only is used for archiving digital objects, but also provides mechanisms for knowledge discovery and efficient access to the holdings in the system [5,6,7].

Information infrastructure focuses on the relationship of a digital object with other data sets from the same discipline or collection. From this perspective, data sets are only useful within the context of an encompassing data collection. Data collections are created by organizing data through the identification of common attributes. The common attributes can be used to create classes of objects for representation in object-oriented databases [7] or can be used to develop schema that define the relationship between objects in relational databases. We have chosen to work with object-relational database management systems that have been tightly coupled to an archival storage system [8] and thus organize common attributes as meta-data within a schema.

However, the schema itself contains information content related to the clustering of attributes into tables, the identification keys that are used to correlate tables, the relational joins that are permitted across the attributes, and the semantics that are used to assign meaning to attributes. This schema meta-data must be quantified to support publication of the collection organization, to support extension of the schema through the addition of new attributes, and to support federation of collections. Hence, information about the organization of a collection is as important as information about digital objects within the collection.

Persistent archives are based upon the concept that both the original digital objects and the information required to assemble the digital objects into a data collection must be archived. Digital objects are not archived as stand-alone entities but, instead, are archived as members of a digital data collection. Persistence is demonstrated by dynamically building the data collection from the individual data objects stored in the archive, dynamically creating the relational joins needed to discover information

within the data collection, and dynamically constructing the presentation interface for the digital objects.

The infrastructure that enables collection based persistent archives can also be used to federate archival storage systems. The ability to migrate a data collection onto new technology needs the same information as that required to federate two collections. Both problems need the ability to interpret how a collection is organized and the ability to dynamically build an information discovery interface into the new collection. A persistent collection can be viewed as the integration of two collections in time (the same collection instantiated on two different sets of technology). A federated collection can be viewed as the integration of two collections in space. In this paper we present the technology that has been implemented to achieve both goals.

II. Information Architecture

The National Partnership for Advanced Computational Infrastructure (a National Science Foundation-funded project led by the San Diego Supercomputer Center) is developing an information architecture to support the creation of scientific data collections. The technologies that are available to build an information infrastructure are:

- Archives—to manage data sets distributed across tertiary storage systems
- Databases—to organize information about the data sets
- Data-handling systems—to provide APIs for access to the data collections
- Digital libraries—to provide services for manipulating and presenting the data collections

The integration of these technologies will lead to a collection-based persistent archive that can be used to support information repositories, supercomputer centers, and digital libraries. We are developing an infrastructure called the "Data Intensive Computing Environment" (DICE) as a first step towards achieving this goal [9,10].

Currently, we are in the process of setting up a general digital library system for ingesting, managing, archiving, and accessing several collections of scientific data whose total size can grow to petabytes with billions of objects. The content of these archives are scientific data sets including documents, images, field-generated data and simulation results for disciplines ranging from astronomy and earth systems science to social science, ecology, and neuroscience. An important issue is to make information in the archived digital libraries available through the web as well as through APIs for processing on supercomputing platforms such as CRAY C90, IBM RS/6000 SP and Tera MTA. We are also investigating automation of the ingestion of data into the digital library. Many of the data sets are produced by remote sensing instruments or are the output of supercomputing applications.

An archival digital library should not only deal with different disciplines but also provide a means of interaction between the disciplines and their collections. This requires a meta-data catalog for schema level attributes such as discipline-specific ontologies and semantics. Also, because scientific data collections are rapidly evolving, one needs to consider the longevity of such ontologies and plan for the ability to migrate them forward in time. Since we are dealing with objects that are to be accessed using different types of APIs and methods, one needs also to migrate forward the methods and procedures that are used in analyzing data. Hence, one needs to go beyond storing preservation-level meta-data for the objects and also consider preservation-level meta-data for methods and APIs.

Our system is built around a Meta-data Catalog (MCAT) developed at SDSC. MCAT is a repository that handles three different levels of meta-data, including:

- Digital object meta-data about type, formats, lineage (creation characteristics), ingestion protocols, usage methods, and domain-specific data set attributes;
- System-level meta-data about audit trails, authentication, access control, and replication and partitioning of data sets; and
- Schema-level meta-data including ontology information for the relationship of the terms in the attribute domain as well as indexing of individual data objects into the ontology.

Digital object meta-data is typically created for every data collection in order to support information discovery. System-level meta-data is used to provide location transparency, access transparency and protocol transparency. Schema-level meta-data is used to provide a way

to migrate the collection to new technology and to federate data collections.

III. MCAT Architecture

The MCAT is a database-based catalog that provides a repository of meta information about digital objects. Digital object attributes are separated into two classes of information within the MCAT:

- System-level meta-data that provides operational information. These include information about resources (e.g., archival systems, database systems, etc. and their capabilities, protocols, etc.), users (e.g., user groups, access and audit control information, privileges, etc.), methods (e.g., server-level and client-level methods and their properties) and data objects (e.g., their formats or types, replication information, location, collection information, etc.).
- Application-dependent meta-data that provides information specific to particular data sets and their collections (e.g., Dublin Core [11,12] values for text objects).

Both of these types of meta-data are extensible, i.e., one can add and/or remove attributes. Internally, MCAT keeps schema-level meta-data about all of the attributes that are defined. The schema-level attributes include:

1. Logical Structure: When a set of meta-data is registered with MCAT, one needs to identify a logical structure in which the rest of the meta-data will be organized. The logical structure should not be confused with database schema and are more general than that. For example, we have implemented the Dublin Core database schema [11] to organize attributes about digitized text. The attributes defined in the logical structure that is associated with the Dublin Core schema contains information about the subject, constraints, and presentation formats that are needed to display the schema along with information about its use and ownership.
2. Attribute Clusters: An attribute cluster is a set of attribute names that are logically interconnected and that have a one-to-one mapping among them. One can view them as a (single or a set of) normalized table(s) in a database context. For example, in the Dublin Core, publisher, name, address, and contact information form a cluster. Contributor name and contributor type form a second cluster; title and its type form yet another cluster, and so on. Similarly in our system-level MCAT core meta-data, we have one cluster for each data replica containing the type, location, and size of the data objects. This aids the implementation of relational joins across the meta-data tables, since each replica has only one value for these properties and these properties provide the physical characteristics of the object. For each cluster, MCAT keeps information about any constraints and comments that can be searched when using the attribute, along with information about use-privileges and grant-of-use-privileges for the cluster. For each attribute, MCAT keeps more than 20 different types of information including its physical, logical and input and output characteristics [13].
3. Token Attributes: Token attributes have a specific function (compared to other attributes); they capture some simple semantic information about the domain of discourse. In the simplest sense, one can use the token attributes to provide the *domain of discourse* for an attribute or a set. One can also use the token attribute to capture semantic translation between discipline domains (e.g., common names vs. scientific names) and also capture hierarchical and equivalence relationships in the domain of discourse. Given the development of semantic standards within a discipline, one can use the token attribute as a bridge between two schemas and provide semantic interoperability.
4. Linkages: Linkages provide a means for interoperating within and between schema. One can define four types of linkages:
   1. attribute-to-attribute,
   2. cluster-to-attribute,
   3. cluster-to-cluster, and
   4. cluster-to-token.

   Each of the linkages can be from one-to-many, many-to-one, or many-to-many. The linkage information is used to generate joins dynamically based on the user's chosen set of attributes. The join algorithm uses Steiner Tree generation of SQL commands from a directed acyclic graph; the DAG is a mapping of clusters and the linkages between them. The linkage information is also used for performing federated

query operation across schemas. The DAG is also used to figure out the notion of an allowed query by disallowing queries that span disjointed graphs.

MCAT provides APIs for creating, modifying and deleting the above structures. The architecture of the MCAT is given in Figure 1. MCAT provides an interface protocol for the application to interact with MCAT. The protocol uses a data structure for the interchange which is called MAPS—Meta-data Attribute Presentation Structure. The data structure, which also has a wire-format for communication and a data format for computation, provides an extensible model for communicating meta-data information. A mapping is being developed to translate from the MAPS structure to the Z39.50 format [14]. Internal to MCAT, the schema for storing meta-data (may possibly) differ from MAPS, and hence mappings between the internal format and MAPS are needed for every type of implementation of the MCAT. Note that it is possible to store the meta-data in databases, flat files, or LDAP directories [15]. MAPS provides a uniform structure for communicating between MCAT servers and user applications.

The MAPS structure defines a query format, an update format and an answer format. The MAPS query format is used by MCAT in generating joins across attributes based on the schema, cluster and linkages discussed above. Depending upon the internal catalog type (e.g., DB2 database, Oracle database, or LDAP) a lower-level target query is generated. Moreover, if the query spans several database resources, a distributed query plan is generated.

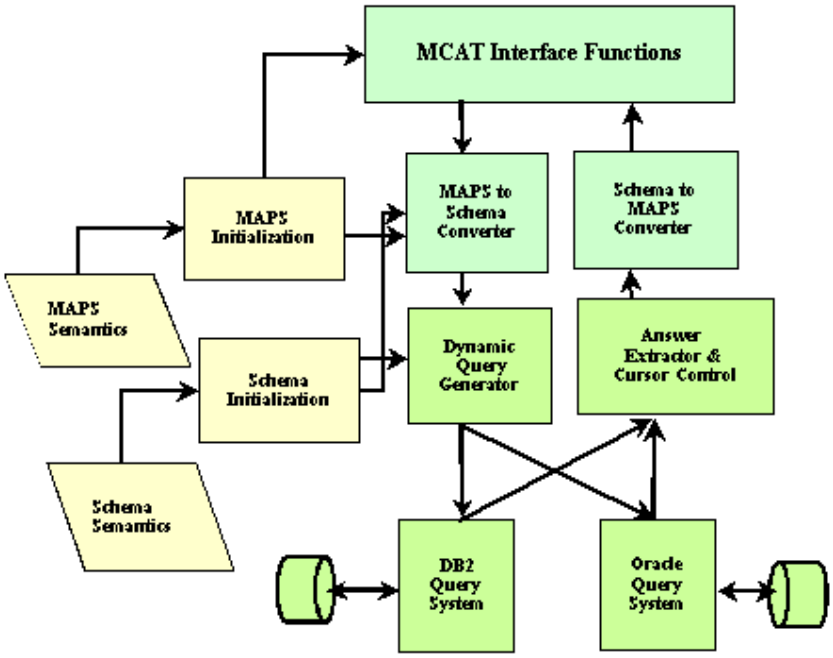

Figure 1. MCAT architecture.

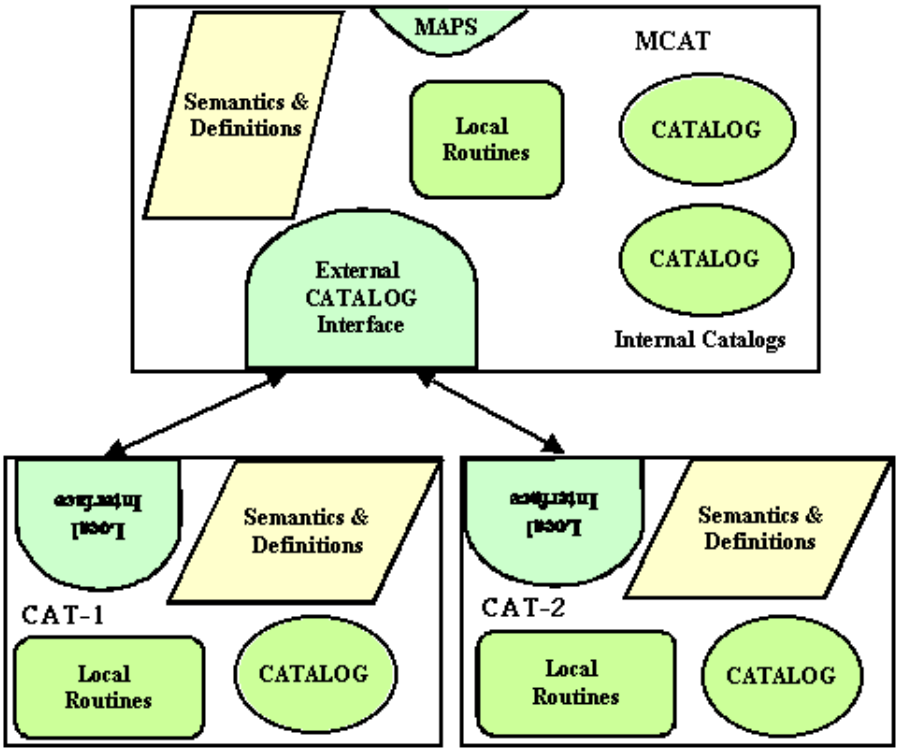

Figure 2. Data collection federation.

Figure 2 shows how multiple data collections can be queried through MCAT. Assume that a group has its own database with a large quantity of meta-data. Moreover, the organization of the data might be peculiar to that group's field of activity and the meta-data may reside as different types of objects—e.g., files, tables, etc. The MCAT-to-catalog interaction is facilitated by a uniform abstraction interface (currently being defined and called the Catalog Interface Definition) that allows external catalogs to communicate with MCAT. The communications would be of two different types: schema-level meta-data communication wherein the semantics of the structure of the external catalog is communicated to MCAT (and vice versa) and meta-data communication where meta-data is transferred between the two catalogs in response to queries and updates. With this definition of an abstraction, including new catalogs would become an exercise in writing middleware components. We believe that the architecture is simple but powerful enough to deal with extensible meta-data schemata and with multiple heterogeneous meta-data services.

The MCAT system supports the publication of schemata associated with data collections, schema extension through the addition or deletion of new attributes, and the dynamic generation of the SQL that corresponds to joins across combinations of attributes. GUIs have been created that allow a user to specify a query by selecting the desired attributes. The MCAT system then dynamically constructs the SQL needed to process the query. By adding routines to access the schema-level meta-data from an archive, it will be possible to build a collection-based persistent archive. As technology evolves and the

software infrastructure is replaced, the MCAT system can support the migration the collection to the new technology. Effectively, the collection is completely represented by the set of digital objects stored within the archive, the schema that contains the digital object meta-data, and the schema-level meta-data that allows the collection to be instantiated from scratch.

IV. SDSC Storage Resource Broker

At SDSC, we have developed a system that provides access to replicated data sets residing in federated and distributed storage systems through meta-data based information and resource discovery. The system consists of two components: the SDSC Storage Resource Broker (SRB) that provides federation and access to distributed and diverse storage resources in a heterogeneous computing environment and the Meta-data Catalog (MCAT) that holds systemic and application or domain-dependent meta-data about the resources and data sets (and users) that are being brokered by the SRB. The SRB provides a uniform API for access to heterogeneous archival storage systems and deals with federation of storage sites and replication of data objects. The MCAT information catalog systems play a vital role in publishing authenticated information, and storing and disseminating the information through a controlled but uniform interface.

The SRB-MCAT system provides a data integration environment that provides

- uniform access APIs across heterogeneous file systems, databases, and archival storage,
- protocol-transparency and location-transparency when accessing distributed systems,
- uniform name space abstraction over the file systems that are being brokered,
- meta-data-based access to files, thus supporting information discovery based on domain and system-dependent meta-information stored along with (or extracted from) the stored files,
- facilities for replication, copying or moving files across heterogeneous systems, performing resource-level operations (proxy operations) on data before delivery to the client: useful for data subsetting, format translation, and other pre-processing applications, and
- an integrated encryption and authentication system that can range from no security to fully encrypted and fully authenticated data transfer including security against man-in-the-middle security intrusions.

There are three main reasons for providing a uniform access mechanism from applications to archival storage system such as HPSS [1] or database-archival storage systems such as DB2-HPSS [8]:

- The data sets under consideration can be very large, making it appropriate to store in archival tape systems directly. Terabyte-sized brain mapping images fall into this category.
- The data sets may be too numerous to be stored in a single file system (and the total size may exceed several gigabytes) making them appropriate to store in a database system.
- The number of data sets may grow with many of the data sets being sparsely used after some initial period of time.

In all the above cases, it may be necessary to hide the fact that the data sets are in archival systems and provide the user with a uniform interface without burdening the user with having to know about the peculiarities of the archival tape systems and the data set pathnames.

The SDSC Storage Resource Broker (SRB) [16,17] is middleware that provides distributed clients with uniform access to diverse storage resources in a heterogeneous computing environment. Storage systems handled by the current release of the SDSC SRB include the UNIX file system, archival storage systems such as UniTree and HPSS, and database Large Objects managed by various DBMSs including DB2, Oracle, and Illustra (Figure 3). Currently, the system runs on supercomputers such as the CRAY C90, CRAY T3E and IBM SP, and on workstations such as Sun, SGI, and DEC platforms. The SRB presents clients with a logical view of data sets stored in the SRB. Similar to the file name in the file system paradigm, each data set stored in SRB has a logical name, which may be used as a handle for data operation. Unlike the file system where the physical location of a file is implicitly implied in its path name through its mount point, the physical location of a data set in the SRB environment is logically mapped to the data sets. Therefore, the actual data of data sets belonging to the same collection may physically reside in different storage systems. A client does not need to remember the physical mapping of

a data set. It is stored as the meta-data associated with the data set in the MCAT catalog. Data sets in the SRB are grouped into a logical (hierarchical) structure called *collections*. The collection provides an abstraction for

- placing similar objects (possibly, physically distributed) under one collection (e.g., image collections of a museum) and
- placing all dissimilar objects that have a common connection under one abstraction (e.g., all the text paragraphs, images, figures, and tables of a document).

The SRB supports data replication in two ways. One can replicate an object during object creation or modification. To enable this, SRB and MCAT allows the creation of *logical storage resources* (LSR) which are a grouping of two or more resources. When an application creates or writes a data set in these logical resources, then the operations are performed on all the resources. The result of using a LSR is that a copy of the data is created in each of the physical resources belonging to the logical resource. It is possible to specify that the write operation is successful if *k* of the *n* copies are created. The user can modify all the copies of the data by writing to the data set with a "write all." However, this operation can lead to an inconsistency if there is a failure in the middle of the operation. The SRB also provides an off-line replication facility using to replicate an existing data set. This operation can also be used for synchronization purposes. When accessing replicated objects, SRB will open the first available replica of the object as given by a list from MCAT. (Using the Network Weather Service [18], one can order this list based on some criteria.)

SRB also provides a facility for resource-side proxy operations. That is, one can define and compile operations that are applied to data sets near the resource before sending the object to the client. This feature can be used for performing system-level operations like copy or move as well as application-specific operations such as data sub-setting, format conversions, and automatic meta-data extraction.

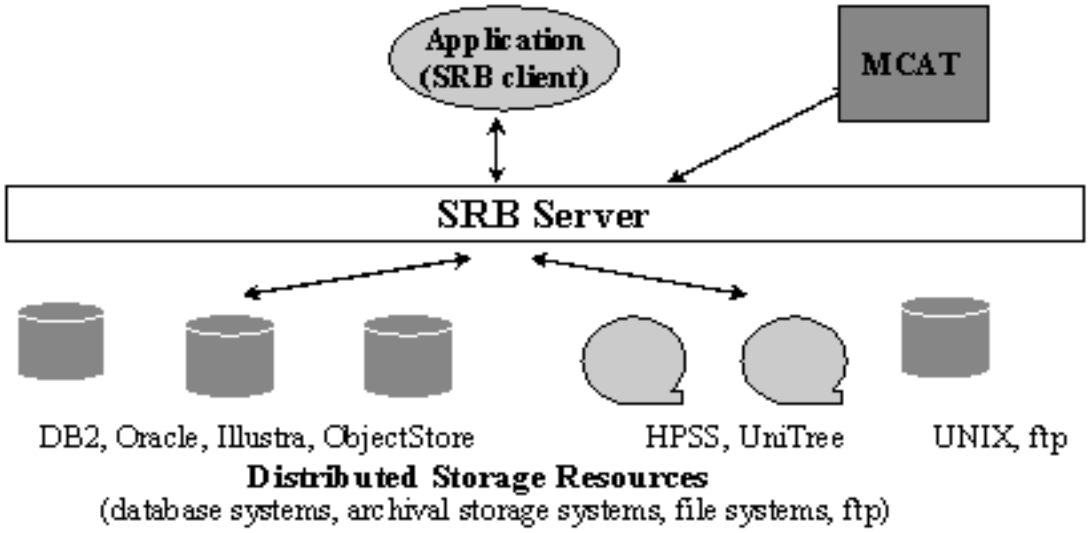

Figure 3. Simplified view of the SRB middleware.

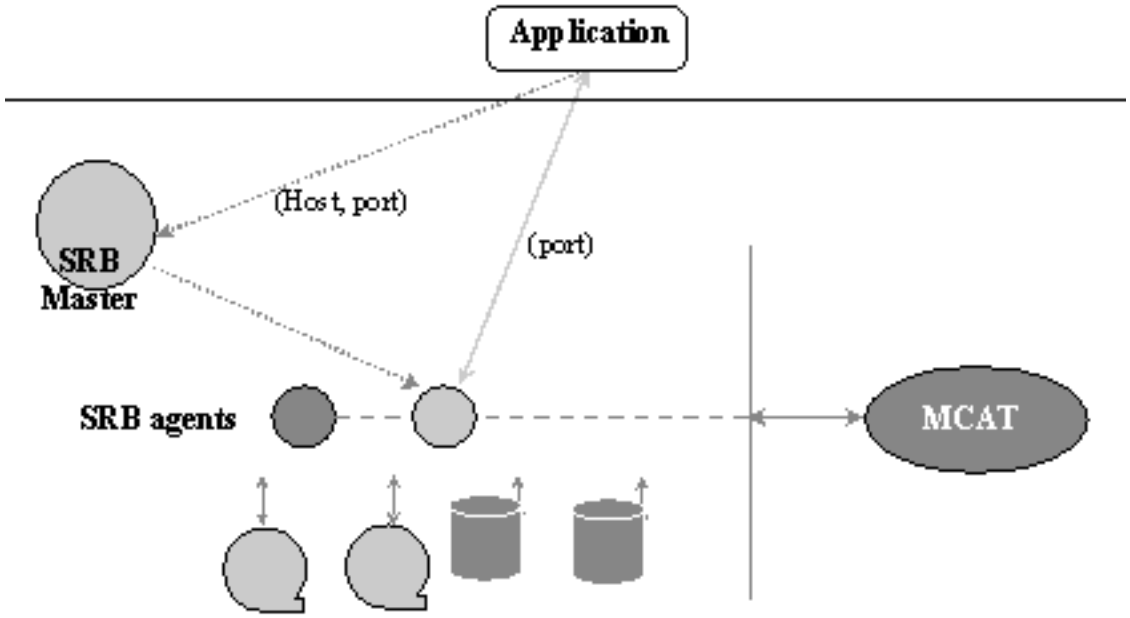

Figure 4. The SRB process model.

The SRB also provides authentication and encryption facilities [19,20], access control list and ticket-based access [21], and auditing capabilities to give a feature-rich environment for sharing distributed data collections among users and groups.

Figure 4 depicts the SRB process model. The design of the SRB server is based on the traditional network connected client/server model but has the additional capability of federation. Once a connection from a client is established and authenticated, a SRB agent is created that brokers all the operations for that connection. A client application can have more than one connection to a SRB server and to as many servers as required. The federation of SRBs implies that a client can connect to any SRB server and can also access a resource that is brokered by any server. An inter-SRB communication protocol supports the federation operation. The SRB communicates with MCAT to obtain meta-information about the data set, which it then uses for accessing the data set.

Summary

The combined SRB-MCAT system support for federation of data collections can also be applied directly to the task of federating archival storage systems. For data collections that span multiple archives, the system is able to build a data collection that physically resides at the multiple locations, but which is accessed through a uniform logical interface. The federation of multiple collections is the spatial equivalent of the task of migrating a collection to new technology. By building the common information management infrastructure needed for both tasks, it then becomes possible to build a collection based persistent archive that is distributed across multiple sites. One can consider a digital library that has data holdings at multiple institutions, linked through the SRB-MCAT infrastructure. Additionally, the software at any one of the institutions can be updated, with the local data collection rebuilt on the new infrastructure through the MCAT schema-level meta-data. The distributed archive can then be maintained as a persistent archive that will be sustainable through an arbitrary number of technology evolution cycles.

Acknowledgements

This work was supported by the National Science Foundation Cooperative Agreement ACI-9619020 and by the National Archives and Records Administration and the Advanced Research Projects Agency/ITO under ARPA Order No. D570, issued by ESC/ENS under contract F19628-96-c-0020.

References

1. The High Performance Storage System (HPSS), http://www.sdsc.edu/HPSS/.
2. IEEE Storage System Standards Working Group (SSSWG) Project 1244, "Reference Model for Open Storage Systems Interconnection, Mass Storage Reference Model Version 5," Sept. 1994.
3. Foster, I., Kesselman, C., "The Grid: Blueprint for a New Computing Infrastructure," Chapter 5, "Data-intensive Computing," Morgan Kaufmann, San Francisco, 1999.
4. Moore, R., "Enabling petabyte computing, The Unpredictable Certainty, Information Infrastructure through 2000," National Academy Press, 1997.
5. Adams, D., Hansen, D., Walker, K., "Scientific Data Archive at the Environmental Molecular Sciences Laboratory," *Proceedings of the 15th IEEE Symposium on Mass Storage Systems*, April 1996.
6. Baru C., Frost, R., Marciano, R., Moore, R., Rajasekar, A., and Wan, M., "Meta-data to support information based computing environments," *Proceedings of the IEEE Conference on Meta-data*, Silver Spring, MD, Sept. 1997.
7. Shiers, J., "Building a Database for the LHC—the Exabyte Challenge," *Proceedings of the 15th IEEE Symposium on Mass Storage Systems*, April 1996.
8. The DB2/HPSS Integration project, http://www.sdsc.edu/MDAS.
9. Baru C. "Archiving Meta-data," 2nd European Conference on Research and Advanced Technology for Digital Libraries (poster), Sept.19-23, 1998, Crete, Greece.
10. Data-Intensive Computing Environment reports, URL http://www.npaci.edu/DICE/.
11. The Dublin Core, http://www.ukoln.ac.uk/meta-data/resources/dc.html.
12. The Warwick Framework, Carl Lagoze, *D-Lib Magazine*, July/August 1996, http://www.dlib.org/dlib/july96/lagoze/o7lagoze.html
13. MCAT – A Meta Information Catalog (V1.1), Technical report: http://www.npaci.edu/DICE/SRB/mcat.html
14. Tomer, C., "Information Technology Standards for Libraries," *Journal of the American Society for Information Science*. 43: 566-570, 1992.
15. Light-Weight Directory Access Protocol (LDAP) implementation by Netscape, http://www.netscape.com/comprod/server_central/-product/directory/index.html
16. Baru, C., *et al.*, "A data handling architecture for a prototype federal application," *Proceedings of the IEEE Conference on Mass Storage Systems*, College Park, MD, March 1998.
17. Baru C., Moore, R., Rajasekar, A., and Wan, M., "The SDSC Storage Resource Broker," *Proceedings of CASCON'98 Conference*, Nov. 30–Dec. 3, 1998, Toronto, Canada.
18. Network Weather Service (NWS), http://www.sdsc.edu/~nspring/nws.html.
19. Schroeder W., "The SDSC Encryption / Authentication (SEA) System," Distributed Object Computation Testbed (DOCT) project white paper, http://www.sdsc.edu/~schroede/sea.html.
20. Schroeder W., "The SDSC Encryption and Authentication (SEA) System," Special Issue of Concurrency: Practice and Experience—Aspects of Seamless Computing, John Wiley & Sons Ltd., 1999.
21. Baru C., and Rajasekar, A., "A Hierarchical Access Control Scheme for Digital Libraries," *Proceedings of the 3rd ACM Conference on Digital Libraries*, Pittsburgh, PA, June 23-25, 1998.