GWD-R
Category: Recommendation
Persistent Archive Research Group

Reagan W. Moore, SDSC
Andre Merzky, ZIB
27-July-2002
Revised 17-February-2003

Persistent Archive Basic Components

Status of This Memo: Recommendations

This memo provides information to the Grid community specifying the minimal capabilities that should be provided by virtual data grids to enable the creation of persistent archives. Distribution is unlimited.

Copyright Notice

Abstract

The preservation of data is a difficult task that requires management of technology evolution and management of authenticity, while ensuring that risk due to catastrophes is minimized. The mechanisms provided by data grids for interoperability across data management technologies provide part of the required infrastructure. The ability to apply archival processes for the generation of archival forms of digital entities is supported by virtual data grids. This paper specifies the minimal capabilities that should be provided by virtual data grids to enable the creation of persistent archives that hold data for lifetimes greater than that of the underlying software technologies.

Contents

1. Persistent Archive Description

The Persistent Archive Research Group of the Global Grid Forum promotes the development of an architecture for the construction of persistent archives. Persistent archives provide mechanisms to manage technology evolution. During the lifetime of the persistent archive, each software and hardware component may be upgraded multiple times. We apply the term “persistent” to the concept of an archive to represent the evolution of the infrastructure over time. The challenge is creating an architecture that maintains the authenticity of the archived collection, while minimizing the effort needed to incorporate new technology. Fortunately, persistent archives are conceptually equivalent to virtual data grids. They both need mechanisms to support access to heterogeneous types of storage systems and information repositories, while supporting the re-creation of derived data products. Hence the persistent archive research group is examining how persistent archives can be built from virtual data grids. Virtual data grids are distributed systems that tie together data management systems and compute resources. Virtual data grids are differentiated from data grids by the ability either to access derived data products or to re-create the derived data products from a process description. Virtual data grids support the application of processes to create derived data products. In the context of persistent archives, virtual data grids are capable of applying the archival processes needed to create archival forms of digital entities.

Data grids provide a logical name space into which digital entities can be registered. The logical name space is used to support global, persistent identifiers for each digital entity. Data grids map distributed state information onto the logical name space for each grid service. Examples are replica information for the location of each physical copy of a digital entity, descriptive metadata that is associated with each digital entity, authenticity attributes associated with each digital entity, etc. Archival processes create the state information that is mapped onto the logical name space. Thus systems for the management of archival processes can be built out of data grid and virtual data grid technologies.

The digital entities that are managed by persistent archives can be quite complex. A digital entity can be a single document, or a compound document with multiple components. A digital entity can also be the collection that is assembled by grouping multiple documents. All of these forms of digital entities can be manipulated by archival processes.

A persistent archive manages collections of digital entities. The collection itself can be thought of as a derived data product that results from the application of archival processes to a group of constituent documents. The collection is used to provide a context for the digital entities that are stored in the archive. Discovery of an individual digital entity is accomplished by querying the descriptive metadata. A description of the processing steps used to create the collection can be archived. Thus the collection is itself a derived data product. A request for access to the collection can result in a query against an instantiated version of the collection, or the execution of the processing steps to instantiate the collection. A persistent archive can be treated as a virtual data grid that manages access to derived data collections, and manages the re-creation of the derived data collections if they are not already instantiated. Persistent archives also manage the migration of data from old storage systems to new storage systems and manage transformative migrations, in which the encoding standard used to describe a data entity is changed to a new standard. Management of the application of transformative migrations again is equivalent to management of derived data products in a virtual data grid.

1.1 Archival Processes

Archivists rely on persistent archives to support archival processes, including appraisal, accessioning, arrangement, description, preservation, access, and re-purposing. A virtual data grid provides support for digital entities, which are either the collection or the digital entities organized by the collection. To demonstrate the capabilities provided by virtual data grids, we examine how these capabilities can be used to implement each of the standard archival processes. For each archival process, we list the corresponding processing steps, and the technologies that enable support for the processing steps.

1.1.1 Appraisal (App.)

Selection of materials to archive – Data grids provide a logical name space that supports the organization of digital entities into collection/sub-collection hierarchies. The logical name space is decoupled from the underlying storage systems, making it possible to sort digital entities without moving them. It is possible to represent a digital entity by a handle (such as a URL), register the pointers into the logical name space, and organize the pointers independently of the data.

1.1.2 Accessioning (Acc.)

Controlled data import – Data grids put digital entities under management control, such that automated processing can be done across an entire collection. Data grids provide mechanisms that can be used to validate data models, extract metadata, and authenticate the identification of the submitter. These capabilities are used to manage the ingestion of digital entities into the archive under process control, and are typically implemented as remote proxies that can be executed at the storage repository that holds the digital entity.

1.1.3 Arrangement (Arr.)

Context management – Data grids use collections to define the context to associate with data entities. The context includes provenance information describing the processes under which the data entities were created, attributes used to support information discovery to identify an individual data entity, and relationships that can be used to determine whether associated attribute values are consistent with implied knowledge about the collection, or represent anomalies and artifacts. An example of a knowledge relationship is the range of permissible values for a given attribute, or a list of permissible values for an attribute. The context management also is used to control the level of granularity associated with the organization of the data entities into collections/sub-collections. Containers, the digital equivalent of a cardboard box, are used to physically aggregate data entities. Data grids also provide support for logical organization of digital entities into collection hierarchies.

1.1.4 Description (Desc.)

Global name space – The ability to identify derived data products is based on persistent logical identifiers that are independent of the local storage system file names. For persistent archives this includes the ability to provide persistent logical identifiers for the data entities stored within the data collections. The global name space may be organized into a collection/sub-collection hierarchy with each sub-collection supporting unique metadata. Each sub-collection is described by an extensible set of attributes that can be defined independently of other sub-collections.

Derived product characterization – Both the derived data products (transformative migrations of digital entities to new encoding formats) and the processes used to generate the derived data products can be characterized in virtual data grids. For persistent archives, the derived data product can be a data collection or the transformative migration of a digital entity. Infrastructure independent representations are used to describe both the derived data product and the processes used to re-create the derived data product. Infrastructure independent representations are typically created by transforming from a proprietary encoding format to a non-proprietary encoding standard such as the eXtensible Markup Language (XML) for metadata and documents, the Hierarchical Data Format (HDF) for binary array data, and an XML Schema and Data Definition Language table structure for collections.

1.1.5 Preservation (Pres.)

Instantiation – A virtual data grid provides the ability to execute a process description. An example is the Chimera system that defines an abstract representation for the steps in the process, and then instantiates the processes as applications running on grid resources. For a persistent archive, this is the ability to instantiate a data collection from its infrastructure independent representation.

Storage repository abstraction – The ability to migrate digital entities between different types of storage systems is provided by data grids through a storage repository abstraction that defines the set of operations that can be performed on a storage system. The heterogeneous storage repositories can also represent different versions of storage systems and databases as they evolve over time. When a new infrastructure component is added to a persistent archive, both the old version and new version will be accessed simultaneously while the data and information content are migrated onto the new technology. Through use of replication, the migration can be done transparently to the users. For persistent archives, this includes the ability to migrate a collection from old database technology onto new database technology.

Disaster recovery – Data grids manage replicas of digital entities, replicas of collection attributes, and replicas of collections. The replicas can be located at geographically remote sites, ensuring safety from local disasters.

Persistency – Virtual data grids provide a consistent environment, which guarantees that the administrative attributes used to identify derived data products always remain consistent with migrations performed on the data entities. The consistent state is extended into a persistent state through management of the information encoding standards used to create platform independent representations. The ability to migrate from an old representation of an information encoding standard to a new representation leads to persistent management of derived data products.

Authenticity – Data grids track all operations done on each data entity, to be able to guarantee that the information and knowledge content of each digital entity have only been transformed by approved archivists. Audit trails record the dates of all transactions and the names of the persons who performed the operations. Digital signatures and checksums are used to verify that between transformation events the digital entity has remained unchanged.

1.1.6 Access (Ac.)

Derived data product access – Virtual data grids provide direct access to the derived data product when it exists. This implies the ability to store information about the derived data products within a collection that can be queried. A similar capability, implemented as a finding aid, is needed to characterize the multiple data collections and contained data entities that are stored in a persistent archive.

Data transport – Data grids provide transport mechanisms for accessing data in a distributed environment that spans multiple administration domains. This includes support for moving data and metadata in bulk, while authenticating the user. Data grids also provide multiple roles for characterizing the allowed operations on the stored data, independently of the underlying storage systems. Users can be assigned the capabilities of a curator, with the ability to create new sub-collections, or annotator with the ability to add comments about the digital entities, or submitter, with the ability to write data into a specified sub-collection, or public user, with the ability to read selected sub-collections.

Storage repository abstraction - Data grids provide the mechanisms needed to support distributed data access across heterogeneous data resources. Data grids implement servers that map from the protocols expected by each proprietary storage repository to the storage repository abstraction. This makes it possible to access digital entities through a standard interface, no matter where it is stored.

1.1.7 Re-purposing (Rep.):

Collection re-creation. The utility and usefulness of archived data is directly proportional to the ability to extract information and knowledge for application in new situations. Re-purposing corresponds to mapping the original context used to describe the digital entities to a new context. The mapping is intended to make the archived material relevant for new uses, beyond the original context under which the archival collection was formed. In one sense, re-purposing of an archival collection corresponds to the execution of the archival tasks of description, arrangement, and access to create a new context for the archived material.

1.2 Persistent Archive Functionality Requirements

The archival processes that have been described need to be mapped onto the functionalities that are provided by data grids. There are multiple challenges in doing the mapping, including the use of different terminology for describing the basic capabilities in data grids and the fact that data grid capabilities are used by more than one archival process. Each data grid capability has been implemented to address a particular data grid functionality requirement. There is no one-to-one mapping between the requirements needed by data grids and the requirements needed by archival processes. We will describe the capabilities that data grids provide using data grid terminology, and then show how these capabilities can be mapped onto the archival processes.

The requirements for a persistent archive can be expressed in general as "transparencies" that hide virtual data grid implementation details. Examples include digital entity name transparency, data location transparency, platform implementation transparency, encoding standard transparency, and authentication transparency or single sign-on systems. The capabilities of a persistent archive can be characterized as the set of “transparencies” needed to manage technology evolution. Implementations exist in data grids for at least four key functionalities or transparencies that simplify the complexity of accessing distributed heterogeneous systems:

- Name transparency – The ability to identify a desired digital entity without knowing its name can be accomplished by queries on descriptive attributes, organized as a collection. Persistent archives are inherently archives of collections of digital entities that map from unique attribute values to a global, persistent, identifier.
- Location transparency – The ability to retrieve a digital entity without knowing where it is stored can be accomplished through use of a logical name space that maps from the global, persistent, identifier to a physical storage location and physical file name. If the data grid owns the digital entities (stored under the data grid user ID), the administrative attributes for storage location and file name can be self-consistently updated every time the digital entity is moved.
- Platform implementation transparency – The ability to retrieve a digital entity from arbitrary types of storage systems can be accomplished through use of a data grid that provides a storage repository abstraction. The data grid maps the protocols needed to talk to the storage systems to the operations defined by the storage repository abstraction. Every time a new type of storage system is added to the persistent archive, a new driver is added to the data grid to map from the new storage access protocol to the data grid data transport protocol. Similar platform transparency is needed for the information repository in which the persistent archive stores the collection context. An information repository abstraction is defined for the set of operations needed to manipulate a catalog in an information repository, or database.
- Encoding standard transparency – The ability to display a digital entity requires understanding the associated data model and encoding standard for information. If infrastructure independent standards are used for the data model and encoding standard (non-proprietary, published formats), a persistent archive can use transformative migrations to maintain the ability to display the digital entities. The transformative migrations will need to be defined only between the infrastructure independent standards.

The infrastructure that supports the above transparencies exists in multiple data grid implementations. When one examines the data grid implementations, it is possible to identify over 150 different capabilities that have been implemented to facilitate the management of data and information in distributed environments. The challenge is defining the minimal set of capabilities that should be provided by a data grid for implementing a viable persistent archive. The fundamental capabilities can be listed as:

- Logical name space
- Storage repository abstraction

- Information repository abstraction
- Distributed resilient scalable architecture
- Virtual data grid

A set of core capabilities has been defined in Table 1. The list includes the essential capabilities that simplify the management of collections of digital entities while the underlying technology evolves. The use of each capability by one of the seven archival processes is indicated. The columns are labeled by App. (Appraisal), Acc. (Accessioning), Arr. (Arrangement), Desc. (Description), Pres. (Preservation), Ac. (Access), and Rep. (Re-purposing).

The table indicates one of the problems in defining a core set of capabilities, in that many of them are used by most of the archival processes. Thus the logical name space is used when referencing archived digital entities by all of the archival processes. This implies there may be a better decomposition of archival tasks that is more strongly aligned with the application of a logical name space versus the mechanisms used to manipulate digital entities. We choose to retain the traditional characterization of archival processes.

Possibly the unique capability that must be present in a persistent archive is the ability to preserve authenticity. This implies an environment in which only authorized actions can take place. Every operation within the persistent archive must be tracked, and the corresponding metadata updated to guarantee consistency of the metadata to preserve authenticity. This can be most easily implemented by having the digital entities owned by the organizing collection. This forces access to be done through the collection, making it possible to track all operations that are done on the digital entities, from transformative migrations, to media migrations, to replication, to accesses.

We note that the choice of core capabilities is an opportunistic definition of the mechanisms that are now available through data grids. We recognize that many of the core capabilities can be implemented as procedural policies on current file system based storage repositories, without using data grid technology. For example, authenticity can be preserved by defining a set of user IDs that are allowed to write to the archive. One can then require that the defined set of persons manually enter characterizations of all operations that they perform. In practice, this approach would be labor intensive. The list of core capabilities is intended to minimize the labor associated with organizing, managing, and evolving a persistent archive.

A question is whether the levels of abstraction associated with virtual data grids are consistent with operations in persistent archives. One can think of a data grid as the set of abstractions that manage differences across storage repositories, information repositories, knowledge repositories, and execution systems. Data grids also provide abstraction mechanisms for interacting with the objects that are manipulated within the grid, including digital entities (logical namespace), processes (service characterizations or application specifications), and interaction environments (portals). The data grid approach can be defined as a set of services, and the associated APIs and protocols used to implement the services. The data grid is augmented with portals that are used to assemble integrated work environments to support specific applications or disciplines. An example is an archivist workbench, which provides separate functions for each of the archival processes. A major question is whether a persistent archive is better implemented as a virtual data grid, incorporating the required functionality directly into the grid, or as a portal, with the required authenticity and management control implemented as an application interface.

Core Capabilities	**App.**	Acc.	**Arr.**	**Desc.**	**Pres.**	**Ac.**	**Rep.**
Storage repository abstraction		x	x		x	x	x
Storage interface to at least one repository		x	x		x		x
Standard data access mechanism		x	x		x	x	x
Standard data movement protocol support		x	x		x	x	x
Containers for data		x	x		x		x
Logical name space	x	x	x	x	x	x	x
Registration of files in logical name space	x	x	x	x	x		x
Retrieval by logical name		x			x	x	x
Logical name space structural independence from physical name space	x	x	x	x	x		x
Persistent handle		x	x	x	x	x	x
Information repository abstraction	x	x	x	x	x	x	x
Collection owned data	x	x	x	x	x	x	x
Collection hierarchy for organizing logical name space		x	x	x			x
Standard metadata attributes (controlled vocabulary)			x	x	x	x	x
Attribute creation and deletion		x	x	x	x		x
Scalable metadata insertion			x	x	x		x
Access control lists for logical name space to control who can see, add, and change metadata			x	x	x	x	x
Attributes for mapping from logical file name to physical file names			x		x		x
Encoding format specification attributes		x		x		x	x
Data referenced by catalog query						x	x
Containers for metadata		x	x	x	x		x
Distributed resilient scalable architecture	x	x	x	x	x	x	x
Specification of system availability		x			x	x	x
Standard error messages		x	x	x	x	x	x
Status checking		x	x	x	x	x	x
Authentication mechanism	x	x	x	x	x	x	x
Specification of reliability against permanent data loss					x		
Specification of mechanism to validate integrity of data and metadata			x		x	x	x
Specification of mechanism to assure integrity of data and metadata		x	x		x		x
Virtual Data Grid		x	x	x	x	x	x
Knowledge repositories for managing collection properties			x	x			x
Application of transformative migration for encoding format		x	x	x	x	x	x
Application of archival processes		x	x	x	x	x	x

Table 1. Core data grid capabilities for implementing a persistent archive

2. Data Grid Implementations

For a persistent archive implementation to be based upon existing data grids, we must demonstrate that the corresponding capabilities are actually present within current data grid environments. To better understand the current status of data grids, we present an analysis of the capabilities that are already provided by production systems. The Global Grid Forum is promoting the development of standards for the implementation of data grids. A survey has been conducted to identify the capabilities that are supported by most data grid implementations. We note that the data grids were used to support distributed data collections, digital libraries, and persistent archive projects.

A comparison has been made between the Storage Resource Broker (SRB) data grid from the San Diego Supercomputer Center, the European DataGrid replication environment (based upon GDMP, a project in common between the European DataGrid and the Particle Physics Data Grid, and augmented with an additional product of the European DataGrid for storing and retrieving meta-data in relational databases called Spitfire and other components), the Scientific Data Management (SDM) data grid from Pacific Northwest Laboratory, the Globus toolkit, the Sequential Access using Metadata (SAM) data grid from Fermi National Accelerator Laboratory, the Magda data management system from Brookhaven National Laboratory, and the JASMine data grid from Jefferson National Laboratory. These systems have evolved as the result of input by user communities for the management of data across heterogeneous, distributed storage resources.

EGP, SAM, Magda, and JASMine data grids support high energy physics data. The SDM system provides a digital library interface to archived data for PNL and manages data from multiple scientific disciplines. The Globus toolkit provides services that can be composed to create a data grid. The SRB data handling system is used in projects for multiple US federal agencies, including the NASA Information Power Grid (digital library front end to archival storage), the DOE Accelerated Strategic Computing Initiative (collection-based data management), the National Library of Medicine Visible Embryo project (distributed data collection), the National Archives Records Administration (persistent archive prototpye), the NSF National Partnership for Advanced Computational Infrastructure (distributed data collections for astronomy, earth systems science, and neuroscience), the Joint Center for Structural Genomics (data grid), and the National Institute of Health Biomedical Informatics Research Network (data grid).

The systems we examine therefore include not only data grids, but also distributed data collections, digital libraries and persistent archives. Since the core component of each system is a data grid, we can expect common capabilities to exist across the multiple implementations. The systems that provide the largest number of features tend to have the most diverse set of user requirements.

The comparison is an attempt at understanding what data grid architectures provide to meet existing application requirements. The capabilities are organized into functional categories, such that a given capability is listed only once. The categories have been chosen based on the need to manage a logical name space, the management of attributes in the logical name space, the storage abstraction for accessing remote storage systems, the types of data manipulation, and the data grid architecture. Since the listed data grids have been in use for multiple years, the features that have been developed represent a comprehensive cross-section of the features in actual use by production systems. The terms used in the comparison are explained in the Glossary

2.1 Common Data Grid Capabilities:

What is most striking is that common data grid capabilities are emerging across all of the data grids. Appendix A lists the common features organized by functional category. Each data grid implements a logical name space that supports the construction of a uniform naming convention across multiple storage systems. The logical name space is managed independently of the physical file names used at a particular site, and a mapping is maintained between the logical file name and the physical file name. Each data grid has added attributes to the name space to support location transparency, file manipulation, and file organization. Most of the grids provide support for hierarchical logical folders within the namespace, and support for ownership of the files by a community or collection ID.

The logical name space attributes typically include the replica storage location, the local file name, and user-defined attributes. Mechanisms are provided to automate the generation of attributes such as file size and creation time. The attributes are created synchronously when the file is registered into the logical name space, but many of the grids also support asynchronous registration of attributes.

Most of the grids support synchronous replica creation, and provide data access through parallel I/O. The grids check transmission status and support data transport restart at the application level. Writes to the system are done synchronously, with standard error messages returned to the user. At the moment, the error messages are different across each of the data grids. The grids have statically tuned the network parameters (window size and buffer size) for transmission over wide area networks. Most of the grids provide interfaces to the GridFTP transport protocol.

The most common access APIs to the data grids are a C++ I/O library, a command line interface and a Java interface. The grids are implemented as distributed client server architectures. Most of the grids support federation of the servers, enabling third party transfer. All of the grids provide access to storage systems located at remote sites including at least one archival storage system. The grids also currently use a single catalog server to manage the logical name space attributes. All of the data grids provide some form of latency management, including caching of files on disk, streaming of data, and replication of files.

3. Persistent Archive Components

Given a consensus on the set of capabilities provided by a data grid, it is possible to identify those capabilities that are relevant to the creation of a persistent archive. In Appendix B, a description of each of the core capabilities is provided. We define the core capability, describe the functionality that would be provided by the capability, and provide a description of how the capability is implemented in current data grid environments. Note that the choice of implementation is arbitrary, with possibly multiple mechanisms used to implement a particular capability.

A major design issue for the creation of persistent archives is the development of an approach in which the digital entities can be preserved in an unchanged format, while still making it possible for future presentation applications to display the digital entity. The challenge is that the encoding format interpreted by future applications will not be the same as the encoding format used to create the original digital entity. Three approaches are being considered within the archival community to resolve this challenge.

1. Migration of digital entities applies an archival process to create an infrastructure independent representation of the digital entity by changing the encoding format to a non-proprietary standard. In the process, the bits of the digital entity must be changed to the standard encoding format. The expectation is that the transformative migration will need to be done at an infrequent interval.

2. Emulation preserves the original digital entity by migrating the presentation application onto new technology. Instead of migrating the digital entity to new encoding formats, the presentation application is migrated to new operating systems. This requires migrating onto new technology all of the applications that were used to create or view each digital entity. The result is a system that preserves the look and feel of the original software, but at the same time makes it very difficult to apply any new techniques to the interpretation of the digital entities.

3. Migration of digital ontologies. Emulation and migration capabilities can be combined by creating a digital ontology that organizes the relationships present within a digital entity. Instead of changing the encoding format of the digital entity to a non-proprietary standard, a digital ontology is created that defines the relationships present within the digital entity. The digital ontology is migrated onto new encoding standards for relationships as they become available. These relationships include the structural relationships that define how to turn the bits into binary arrays, or words, or tables. Logical relationships are used to apply semantic tags to the structures. Spatial relationships are used to map binary arrays to coordinate systems. Temporal relationships are used to apply time stamps to structures. The digital ontology specifies the order in which the relationships need to be applied to correctly interpret the information and knowledge content. The digital entity can be kept in its original encoding format. The presentation application is emulated as the set of operations that can be performed on the defined relationships. The set of operations can be kept fixed on the original set, or they can be expanded over time as new capabilities or created (such as causal queries on time stamps). In effect, the presentation application is emulated as operations on a digital ontology, and the digital ontology is migrated forward in time onto new encoding formats.

All references to migration in this report can be interpreted as either migration of digital entities onto new encoding formats for display by future applications, or migration of digital ontologies onto new encoding formats for display through a standard set of operations.

3.1 Example Persistent Archive

An example persistent archive has been constructed using the San Diego Supercomputer Center Storage Resource Broker data grid. The persistent archive components based upon the SRB include:

- Logical name space implemented in the Metadata Catalog (MCAT). The logical names are chosen by the archivist. A mapping is maintained from the logical name to the physical file location. Distributed state information is mapped onto the logical name space as attributes. The logical names are infrastructure independent, and are organized in a collection hierarchy, allowing the specification of different descriptive metadata for each sub-collection. Soft links and shadow links are supported for the logical organization and registration of digital entities. Digital entities may include files, URLs, SQL command strings, directories, database tables.
- Storage repository abstraction implemented in the SRB. The set of operations that are supported include Unix file system operations (create, open, close, unlink, read, write, seek, sync, stat, fstat, mkdir, rmdir, chmod, opendir, closedir, and readdir), latency management operations (aggregation of data, I/O commands, and metadata), and metadata manipulation (extraction, registration) through use of remote proxies. Containers are used to physically aggregate digital entities before storage into archives. Both digital entities and containers can be replicated. The storage repository abstraction is used to manage data within Unix file systems, archives, object-relational databases, object ring buffers, storage resource managers, FTP sites, GridFTP sites, and Windows file systems.
- Information repository abstraction implemented in the MCAT. Mechanisms are supported for schema extension through addition of new attributes, table restructuring, and metadata import and export through XML files. Soft links are supported for logical reorganization of digital entities within a collection hierarchy. Metadata attributes are maintained for provenance attributes (Dublin core), administrative metadata (file location), descriptive metadata (user-defined attributes), and authenticity metadata (audit trails, digital signatures). The information repository abstraction is used to manage metadata in both proprietary and non-proprietary databases including DB2, Oracle, Sybase, Informix, SQLServer, and PostgresSQL.
- Distributed resilient architecture implemented through a federated client server architecture.

Servers are installed in front of each storage repository and in front of the information repository. Access to the system results in the creation of a service instance that manages further interactions for the request. The service instance retrieves all required distributed state information from the MCAT catalog that is needed to complete the request, and interacts with remote servers as needed to access the data. The system has been designed to minimize the number of message sent over wide area networks to improve performance and increase reliability. Data retrieval requests are automatically retried on a replica when a storage repository does not respond. All error messages generated by the network, storage repository, and information repository are returned to the user. Consistency constraints on distributed state information are explicitly integrated into the software through use of write locks and synchronization flags. This makes it possible to write to a file that has been aggregated into a container and replicated into an archive, lock out competing activities to avoid over-writes, and then synchronize all replicas to the new state. All system metadata is automatically generated and updated by the SRB on each request.

– Virtual data grid implemented through use of remote proxies and external process management systems. The SRB provides a mechanism to process data remotely, before it is sent over a network. The Ohio State University DataCutter technology is used to filter data. External process management systems can control the generation of derived data products through application of remote proxies or the DataCutter software. Interactions with databases can be expressed through SQL command strings that are registered into the logical name space. The SRB is able to apply simple transformative migrations such as unit conversion and reformatting of query results into HTML or XML. More complex transformations require the use of a process management system.

4. Summary

A proposed set of core capabilities can be defined for minimizing the labor required to implement, manage, and evolve a persistent archive. The capabilities are present within implementations of current data grids. Many of the capabilities are general properties that have been implemented across almost all existing data grids. A characterization of each capability has been defined. This characterization can be used as the set of requirements for defining a persistent archive architecture.

5. Author Information

Reagan W. Moore
San Diego Supercomputer Center (SDSC)
9500 Gilman Drive, MC-0505
La Jolla, CA 92093-0505]
moore@sdsc.edu

Andre Merzky
Zuse Institute Berlin (ZIB)
Scientific Visualization
Takustr. 7 - D-14195 Berlin
merzky@zib.de

6. Acknowledgements

The results presented here were supported by the NSF NPACI ACI-9619020 (NARA supplement), the NSF NSDL/UCAR Subaward S02-36645, the DOE SciDAC/SDM DE-FC02-01ER25486 and DOE Particle Physics Data Grid, the NSF National Virtual Observatory, the NSF Grid Physics Network, and the NASA Information Power Grid. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the National Science Foundation, the National Archives and Records Administration, or the U.S. government.

The data grid and toolkit characterizations were only possible through the support of the following persons: Igor Terekhov (Fermi National Accelerator Laboratory), Torre Wenaus (Brookhaven National Laboratory), Scott Studham (Pacific Northwest Laboratory), Chip Watson (Jefferson Laboratory), Heinz Stockinger and Peter Kunszt (CERN), Ann Chervenak (Information Sciences Institute, University of Southern California), Arcot Rajasekar (San Diego Supercomputer Center).

More information about the example persistent archive based on the San Diego Supercomputer Center Storage Resource Broker can be found at http://www.npaci.edu/DICE/SRB/index.html and http://www.sdsc.edu/NARA/.

7. Glossary

Terms used to describe data grid capabilities are defined below.

7.1 Logical name space

- A logical name space is a naming convention for labeling digital entities. The logical name space is used to create global, persistent identifiers that are independent of the storage location.
- Within the logical name space, information consists of semantic tags that are applied to digital entities.
- Metadata consists of the semantic tags and the associated tagged data, and is typically managed as attributes in a database. Metadata is called data about data.
- Collections organize the metadata attributes that are managed for each digital entity that is registered into the logical name space
- Registration corresponds to adding an entry to the logical name space, creating a logical name and storing a pointer to the file name used on the storage system.
- The logical name space can be organized as a collection hierarchy, making it possible to associate different metadata attributes with different sets of digital entities within the collection. This is particularly useful for accession, arrangement, and description.
- Logical folders within a collection hierarchy represent sub-collections, and are equivalent to directories in a file system, but are used to manage different sets of metadata attributes.
- Soft links represent the cross registration of a single physical data object into multiple folders or sub-collections in the logical name space
- Shadow links represent pointers to objects owned by individuals. They are used to register individual owned data into the logical name space, without requiring creation of a copy of the object on storage systems managed by the logical name space.
- Replicas are copies of a file registered into the logical name space that may be stored on either the same storage system or on different storage systems.
- Collection-owned data is the storage of digital entities under a Unix user ID that corresponds to the collection. Access to the data is then restricted to a server running under the collection ID. User access is accomplished by authentication to the data grid, checking of access controls for authorization, and then retrieval of the digital entity by the data grid from storage through the collection ID for transmission to the user.

7.2 Storage repository abstraction

- A storage repository is a storage system that holds digital entities. Examples are file systems, archives, object-relational databases, object-oriented databases, object ring buffers, FTP sites, etc.
- A storage repository abstraction is the set of operations that can be performed on a storage

repository for the manipulation of data.
- A container is an aggregation of multiple digital entities into a single file, while retaining the ability to access and manipulate each digital entity within the container.
- Load balancing within a logical name space consists of distributing digital objects across multiple storage systems
- Storage completion at the end of a single write corresponds to synchronous data writes into storage.
- Third party transfer is the ability of two remote servers to move data directly between themselves, without having to move the data back to the initiating client
- Metadata about the I/O access pattern is used to characterize interactions with a digital entity, recording the types of partial file reads, writes, and seeks.
- Synchronous updates correspond to finishing both the data manipulations and associated metadata updates before the request is completed.
- Asynchronous updates correspond to completion of a request within the data handling system, after the return was given to a command.
- Storage Resource Managers control the load on a Hierarchical Resource Manager or disk file system. They rearrange the submitted work load to optimize retrieval from tape, stage data from the HRM to a disk cache, and manage the number of allowed simultaneous I/O requests.

7.3 Information repository abstraction

- An information repository is a software system that is used to manage combinations of semantic tags (attribute names) and the associated attribute data values. Examples are relational databases, XML databases, Lightweight Directory Access Protocol servers, etc.
- An information repository abstraction is the set of operations that can be performed on an information repository for the manipulation of a catalog or collection.
- Template based metadata extraction applies a set of parsing rules to a document to identify relevant attributes, extracts the attributes, and loads the attribute values into the logical collection.
- Bulk metadata load is the ability to import attribute values for multiple objects registered within the logical name space from a single input file.
- Curation control corresponds to the administration tasks associated with creating and managing a logical collection

7.4 Distributed resilient scalable architecture

- Federated server architecture refers to the ability of distributed servers to talk among themselves without having to communicate through the initiating client.
- GSI authentication is the use of the Grid Security Infrastructure to authenticate users to the logical name space, and to authenticate servers to other servers within the federated server architecture
- Dynamic network tuning consists of adjusting the network transport protocol parameters for each data transmission to change the number of messages in flight before acknowledgements are required (window size) and the size of the system buffer that holds the copy of the messages until the acknowledgement is received.
- SDLIP is the Simple Digital Library Interoperability Protocol. It is used to transmit information for the digital library community

7.5 Virtual data grid

- The automation of the execution of processes is managed in virtual data grids. References to the result of a process can result in the application of the process, or direct access to the result.
- Knowledge corresponds to relationships between attributes, or to relationships that characterize properties of a collection as a whole. Relationships can be cast as inference

rules that can be applied to digital entities. An example is the set of structural relationships used to parse metadata from a digital entity in metadata extraction.
- The application of processes at remote storage systems is accomplished through systems such as the DataCutter, a data filtering service developed by Joel Saltz at the Ohio State University, which is executed directly on a remote storage system.
- Transformative migrations correspond to the processing of a digital entity to change its encoding format. The processing steps required to implement the transformative migration can themselves be characterized and archived, and then applied later.
- Digital ontologies organize the set of semantic, structural, spatial, temporal, procedural, and functional relationships that are present within a digital entity. The digital ontology specifies the order in which the relationships need to be applied in order to correctly display or manipulate the digital entity.

8. Intellectual Property Statement

9. Full Copyright Notice

10. References

1. R. Moore, A. Rajasekar, “Common Consistency Requirements for Data Grids, Digital Libraries, and Persistent Archives”, submitted to 12th High Performance Distributed Computing conference, Seattle, Washington, Jun 2003

2. R. Moore, C. Baru, “Virtualization Services for Data Grids”, Book chapter in "Grid Computing: Making the Global Infrastructure a Reality", John Wiley & Sons Ltd, 2003.

3. R. Moore, “The San Diego Project: Persistent Objects”, Proceedings of the Workshop on XML as a Preservation Language, Urbino, Italy, October 2002.

4. Moore, R., C. Baru, A. Rajasekar, B. Ludascher, R. Marciano, M. Wan, W. Schroeder, and A. Gupta, “Collection-Based Persistent Digital Archives – Parts 1& 2”, D-Lib Magazine, April/March 2000, http://www.dlib.org/

5. Moore, R. (2000a), “Knowledge-based Persistent Archives,” Proceedings of La Conservazione Dei Documenti Informatici Aspetti Organizzativi E Tecnici, in Rome, Italy, October, 2000.

6. Moore, R., C. Baru, A. Rajasekar, R. Marciano, M. Wan: Data Intensive Computing, In ``The Grid: Blueprint for a New Computing Infrastructure'', eds. I. Foster and C. Kesselman. Morgan Kaufmann, San Francisco, 1999.

7. Baru, C., R, Moore, A. Rajasekar, M. Wan, "The SDSC Storage Resource Broker,” Proc. CASCON'98 Conference, Nov.30-Dec.3, 1998, Toronto, Canada.

8. Baru, C., R. Moore, A. Rajasekar, W. Schroeder, M. Wan, R. Klobuchar, D. Wade, R. Sharpe, J. Terstriep, (1998a) “A Data Handling Architecture for a Prototype Federal Application,” Sixth Goddard Conference on Mass Storage Systems and Technologies, March, 1998.

9. MCAT - “The Metadata Catalog”, http://www.npaci.edu/DICE/SRB/mcat.html

10. SRB - “The Storage Resource Broker Web Page, http://www.npaci.edu/DICE/SRB/

Appendix A. Data Grid Capability Summary:

A consensus on the approach towards building data grids can be gathered by examining which features are implemented by at least five of the seven surveyed data grids. Across the eleven categories of capabilities covered by the comparison, the following capabilities represent a standard approach. The number of grids that provided a given feature is listed in parentheses, with the default value being all of the grids.

Logical name space	
	Logical name space independence from physical name space
	Hierarchical logical folders (5)
	Management of attributes used for each capability (registration, deletion)
	Deletion of entities from logical name space
	Soft links between objects in logical folders (6)
	Support for collection owned data (5)
	Registration of files into logical name space
Logical name space attributes	
	Replica storage location, local file name
	Group access control lists (5)
	Bulk asynchronous load of attributes (5)
	User defined attributes (5)
Attribute manipulation	
	Automated size, time stamp
	Synchronous attribute update
	Asynchronous annotation (6)
Data Manipulation	
	Synchronous replica creation (6)
Data Access	
	Parallel I/O support (6)
	Transmission status checking (6)
	Transmission restart at application level
	Synchronous storage write
	Standard error messages
	Thread safe client (5)
	Static network tuning (5)
	GridFTP support (5)
Access APIs	
	C++ I/O library API (5)
	Command line interface
	Java interface (6)
	Web service interface (5)
Architecture	
	Distributed client server
	Federated server (6)
	Distributed storage system access
	Third party transfer (5)
	GSI authentication (5)
Latency Management	
	Streaming (6)
	Caching

	Replication (6)
	Staging (5)
System Support	
	Storage Resource Manager interface (5)
	Archive interface to at least one system
	Single catalog server (6)
	Performance for import/export of files greater than 20 files per sec (5)
	Management of file transfer errors (5)

Appendix B. Definition of Core Capabilities for Persistent Archives

1. Storage repository abstraction

Core capability definition:
The set of operations that can be used to manipulate data within a storage repository.

Functionality provided by the capability
A storage repository holds digital entities. By mapping from the storage repository abstraction to the protocols required by a particular storage repository, it is possible to manage data in any type of storage system, including file systems, hierarchical storage managers, databases. To add a new type of storage system to the data grid, a new driver is written to map from the storage repository abstraction to the new access protocols.

Example grid implementation
A standard set of operations for management of distributed data may include Unix file system operations (create, open, close, unlink, read, write, seek, sync, stat, fstat, mkdir, rmdir, chmod, opendir, closedir, and readdir), latency management operations (aggregation of data, I/O commands, and metadata), and metadata manipulation (extraction, registration).

2. Storage interface to at least one repository

Core capability definition
Every persistent archive will contain at least one storage system for holding digital entities.

Functionality provided by the capability
The storage repository is intended to provide long term residency for data, the bits that comprise the digital entity. The information and knowledge content within the data may be annotated and encapsulated in Open Archival Information System (OAIS) packages. While this content may also reside in the storage repository, support for discovery based on the information content would be supported by an information repository.

Example grid implementation
Traditional long term residency systems for data are based on use of tape, managed by a hierarchical storage manager. However the capital cost of disk systems is starting to approach that of tape systems, with similar capacities. By using data grids to manage the digital entities, the user authentication, and the user authorization, the labor costs of disk systems can also be reduced to that of tape.

3. Standard data access mechanism

Core capability definition
The user interface used to access digital entities residing in a storage repository.

Functionality provided by the capability
A standard data access mechanism provides a uniform interface to digital entities residing in the storage repositories. The choice of standard access mechanism can be made separately from the choice of storage repository. The data access mechanism can be kept the same across multiple versions of storage repositories over time.

Example grid implementation
The choice for standard data access mechanism can be a web services interface based on the Open Grid Services Architecture, or a web browser interface. The standard data access mechanism specifies the user Application Programming Interface (API) that will be preserved over time.

4. Standard data movement protocol support

Core capability definition
The data transfer protocol that is used between the API and the storage repository abstraction.

Functionality provided by the capability
A standard data transport protocol minimizes the amount of effort needed to implement a data grid. The transport mechanism should provide parallel I/O support for bulk data transport, reliable and guaranteed delivery of data, and interoperate with a standard authentication protocol.

Example grid implementation
An emerging standard is the GridFTP protocol for data movement. Other protocols are in extensive use, including the SRB data transport and http. It is possible to build protocol conversion mechanisms to map between multiple data transport protocols.

5. Containers for data

Core capability definition
An aggregation mechanism to keep multiple digital entities in a single file.

Functionality provided by the capability
Containers make it possible to guarantee that multiple digital entities are stored on the same media. Containers also provide a needed management function for hierarchical resource managers, by minimizing the number of names that must be maintained in the HRM. Containers provide a latency management function, making it possible to move many digital entities as a single file.

Example grid implementation
The Storage Resource Broker data grid uses containers to aggregate data before storage on HRMs. References to a digital entity within a container causes the container to be cached on disk, off-loading I/O commands from the HRM. A containerization service is being developed as part of the Globus tool kit.

6. Logical name space

Core capability definition
Naming convention for labeling digital entities.

Functionality provided by the capability
The logical name space is used to create global, persistent identifiers that are independent of the storage location. This makes it possible to reference digital entities that reside on multiple storage systems using a common set of names.

Example grid implementation
Replica catalogs provide a mapping from the physical file name to a global identifier. The name space can be organized independently of the directory structures on the storage systems. Global names are created that can be used as persistent identifiers.

7. Registration of files in logical name space

Core capability definition
Mechanism to add digital entities to the name space.

Functionality provided by the capability

A logical name space can be used to register arbitrary digital entities. The most common digital entity used in preservation is a file. Registration corresponds to adding an entry to the logical name space, creating a logical name and storing a pointer to the physical file and its location.

Example grid implementation
Logical name spaces in data grids have been used to register files, URLs, SQL command strings, processing templates for metadata extraction, executables, etc. It is possible to register a link within the logical name space (soft link), which provides a way to re-purpose data without having to replicate the digital entity.

8. Retrieval by logical name

Core capability definition
Mechanism to retrieve a digital entity by mapping from the logical name to the physical location where the digital entity resides.

Functionality provided by the capability
Retrieval can include the physical transport of the digital entity to the client application that made the request. Retrieval can also include the invocation of a display or presentation application. Retrieval can also include the invocation of the digital entity in the case of a URL, or the execution of the digital entity in the case of a SQL command string.

Example grid implementation
Data grids invoke different retrieval mechanisms depending upon the type of user interface or API. Web browsers tend to invoke display applications on retrieval, while C library calls usually process the digital entity using Unix file operations. The digital entity can be partially returned, in the case of partial file reads, or may be returned in its entirety.

9. Logical name space independence from physical name space

Core capability definition
The organization of the logical name space has no dependence upon the organization of the physical name space.

Functionality provided by the capability
The logical name space can be organized as a directory hierarchy or a hierarchical collection. Pointers are used to identify the location of the digital entity within a storage system. The pointers can point to an arbitrary physical directory. For replicas of digital entities, the location of each replica in a storage system can be specified independently of all other replicas.

Example grid implementation
The ability to decouple the logical name space completely from physical names for digital entities makes it possible to manage a wide variety of digital entities. Logical name space independence is particularly important when registering URLs as replicas or each other. Each URL points to a different site, but is recorded as being equivalent. Replica catalogs also require logical name space independence when registering files as replicas of each other, when the files reside in different types of file systems (Windows NT versus Linux).

10. Persistent handle

Core capability definition
Infrastructure independent naming convention for a digital entity. The naming convention can be semantic free as well as location independent.

Functionality provided by the capability

Digital entity names can be persistent if their semantic meaning is decoupled from the physical location representing their storage location. Depending upon the management policies, the logical semantic tag can be kept invariant as the digital entity is migrated across multiple storage systems. Within the context of the collection for which the logical name is created, the naming convention can represent a globally unique identifier. A persistent handle can be implemented as a logical name. The choice of the syntax for the logical name is arbitrary. It can be defined via a handle system relative to an organization, or can be defined relative to a collection, or can be a user-defined name.

Example grid implementation
The implementation of persistent handles requires the ability to manage the consistency of the logical name space. While the persistent handle is held invariant over time, the archival state information mapped to the handle needs to remain consistent. Every operation on digital entities within the logical name space needs to be mirrored by appropriate changes to the location attributes associated with the logical name. Updates in distributed environments can be automated if the digital entities are owned by the collection in which the location attributes are maintained. Then it is possible to guarantee consistency of the persistent handles. The physical file name that represents a digital entity can be kept consistent with the persistent handle. Data grids have been implemented in which digital entities are owned by the collection (consistent environments), and in which digital entities are owned by individuals (consistency dependent upon user policies for updating the replica catalog references).

11. Information repository abstraction

Core capability definition
The set of operations that can be used to manipulate a catalog within an information repository such as a database.

Functionality provided by the capability
Technology evolution applies equally well to information repositories as it does to storage repositories. An abstraction for catalog manipulation operations is needed to make it possible to migrate the persistent archive metadata to new database technology.

Example grid implementation
Typical operations that are performed on information repositories include schema extension, bulk metadata loading, automated SQL generation, bulk metadata extraction and formatting, For a virtual data grid, in which the archived collection context is dynamically created by parsing the digital entities for information content, the information repository abstraction needs to include the ability to dynamically create a database instance.

12. Collection owned data

Core capability definition
Ownership of digital entities within storage repositories by the organizing collection.

Functionality provided by the capability
To maintain authenticity, all manipulations of digital entities within a persistent archive need to be audited. Tracking mechanisms can be built into the policy management specifications for a persistent archive, or they can be integrated into the data management system such that any change to the location or format of a digital entity is automatically recorded in the collection metadata. To minimize manual labor requirements, automation of metadata tracking is required. This can be accomplished by having the collection own the digital entities, requiring the involvement of the collection software before any operation can be performed on the digital entities.

Example grid implementation

Support for collection owned data is becoming a standard capability within data grids. Of the seven grids surveyed, five grids supported collection owned data. An implication is the need for the management of authorization mechanisms to restrict access. In the typical scenario, a user authenticates herself to the data grid. The data grid authenticates itself to the remote storage system, and checks its own access control lists to determine whether the user can manipulate the digital entity. Data grids decouple the management of the users and their access restrictions from the storage repositories. This simplifies administration of storage repositories that hold digital entities for the persistent archive.

13. Collection hierarchy for organizing logical name space

Core capability definition
Use of collection/sub-collection hierarchies for organizing the logical name space attributes used to control digital entities

Functionality provided by the capability
Logical name spaces inherently require the specification of attributes to manage information about the physical location of each digital entity. Additional attributes are used to manage soft links and sub-collection specific metadata. Since each sub-collection can have a different set of attributes, a collection/sub-collection hierarchy is used to organize the logical name space. A more general structure for organization of the logical name space would be a graph, in which relationships are used on each link to define a context for organizing metadata attributes. Such organization mechanisms will be required in the future when knowledge relationships are managed for persistent archives, in addition to the informational semantic tags.

Example grid implementation
Management of a collection hierarchy can be facilitated by the use of schema indirection. The attributes assigned to a collection can be specified by use of two arrays, one to record the attribute name, and one to record the attribute value. The use of a collection hierarchy can also be expressed as the use of schema indirection for organizing attributes. Data grids have been implemented that use explicitly defined tables for digital entity attributes, and that use schema indirection to manage the attributes. Explicitly defined tables are preferred for collections that manage millions of files.

14. Standard metadata attributes (controlled vocabulary)

Core capability definition
Use of standard metadata semantic names for describing collection specific attributes

Functionality provided by the capability
When a collection hierarchy is used to organize attributes for each collection, it is very easy to create semantic terms for the information content that are unique to the sub-collection. By using standard metadata attributes, the utility of the information content can be extended to terms that are in common across multiple collections. Examples are the use of Dublin Core attributes to specify provenance. The associated attribute values also may have embedded semantics. Use of controlled vocabularies is needed to provide a consistent interpretation to both the semantic meaning of an attribute name and the semantic meaning of the attribute value.

Example grid implementation
Data grid collection hierarchies have been organized by inheriting metadata attributes from standard metadata schema, by inheriting metadata from the parent collection, and by assigning unique metadata. When such hierarchies are queried at the top level, it is not uncommon to find hundreds to thousands of metadata attributes across all sub-collections. The use of standard metadata attributes is essential to avoid semantic name explosion.

15. Attribute creation and deletion

Core capability definition
Both attribute names and attribute value assignments can be created and deleted relative to the logical name space.

Functionality provided by the capability
Schema extension is needed to allow collections to evolve over time. The choice of the appropriate context to use for discovery typically depends upon the user community. The semantic names used for discovery will change as the user community evolves. The infrastructure for managing technology evolution must also manage the evolution of the archival collection. This process is usually associated with re-purposing of archival content.

Example grid implementation
Data grids support attribute creation through multiple mechanisms: synchronously when digital entities are registered, asynchronously after digital entities have been registered, and through bulk metadata registration. Attribute values are loaded from externally defined XML files, or through application of templates that apply parsing rules to annotate and extract semantic content. Attribute deletion is done either by setting a flag, or by actual deletion of the attribute from the collection.

16. Scalable metadata insertion

Core capability definition
Mechanisms to automate creation and loading of metadata attribute names and associated values.

Functionality provided by the capability
Scalability is achieved through the automation of metadata manipulation processes. The processes include metadata extraction from the digital entities, the aggregation of the metadata into XML files, and the bulk loading of attributes into metadata catalogs. Scalability is enhanced by the implementation of each of these processes as parallel I/O streams.

Example grid implementation
Scalable metadata insertion is typically achieved by working with parallel database technology that can handle multiple simultaneous insertion streams. This requires parallel technology to generate and manage the parallel I/O streams, either through creation of thread-safe clients, or by the spawning of multiple processes that simultaneously generate the I/O streams.

17. Access control lists for logical name space to control who can see, add, and change metadata

Core capability definition
Mechanisms for managing user authorization for access to persistent archive holdings

Functionality provided by the capability
For persistent archives that implement collection ownership of data held in storage repositories, a mechanism is needed to decide which digital entities can be accessed by each user. This requires an authentication mechanism to identify users, and an authorization mechanism to define access controls. Each collection and each digital entity may need separate access controls. In addition, separate access roles are needed for the collection curator to manage metadata, the collection creator for adding digital entities, and the public for access to the material.

Example grid implementation

Data grids address authorization through the use of access control lists on groups of users and on individual users. The authorization mechanism can be implemented as metadata that is managed about users and user groups. The metadata can be implemented directly within the collection as a collection-specific table, or can be implemented in a separate authorization server that is used to control access to multiple collections. The choice of access roles can include: curator, owner, writer, annotator, and reader.

18. Attributes for mapping from logical file name to physical file names

Core capability definition
The logical name space manages attributes for mapping from the logical name for a digital entity to the physical name under which a digital entity is stored.

Functionality provided by the capability
The logical name space mapping to a physical name space can be one-to-one, with a single digital entity corresponding to the logical name. The mapping can be one-to-many, with multiple replicas associated with a single logical name. The mapping can be one-to-many with semantically equivalent, but syntactically different digital entities associated with the logical name. The mapping can be associative, with digital entities physically aggregated into a container and the container stored in a repository. The attributes in each case can contain not only the location of the digital entity and its name on the remote storage system, but also the name of the protocol required to talk to that storage system, the type of the digital entity, and Unix system semantics for information about the size, ownership, creation date, update date, etc.

Example grid implementation
The set of attributes associated with each digital entity can be unique in data grid implementations. Note that the attributes associated with a replica must be unique to that replica, since it resides at a different storage location or under a different path name on the same storage system. Similarly, it is possible to let the create and update times for each replica be unique, making it possible to track consistency across replicas. By allowing the data type of the digital entity to vary across replicas it is possible to manage syntactically different versions of the same logical digital entity.

19. Encoding format specification attributes

Core capability definition
Specification of the data type or data model associated with each digital entity, or the digital ontology that organizes the relationships present within a digital entity.

Functionality provided by the capability
Since a persistent archive must allow the evolution of the encoding format of each digital entity, an attribute that is managed by the persistent archive should be the data type or data model. When transformative migrations are performed on the digital entity, semantically equivalent replicas are made, that are differentiated by the syntax associated with the new encoding format. The ability to specify the encoding format needs to apply to replicas of digital entities. Alternatively, a digital ontology can be used to characterize the digital entity, with transformative migrations on the encoding format for relationships applied to the digital ontology. The set of operations that can be performed on the defined relationships will also need to be characterized, and emulated by future presentation applications.

Example grid implementation
Encoding format specification is used for both static transformative migrations and dynamic transformative migrations. The conversion from an old encoding format to a new encoding format can be thought of as a static transformation that is performed once. The conversion from an encoding format to an associated display can be thought of as a dynamic transformation that is invoked every time the digital entity is viewed. Graphical user interfaces to data grids typically

access the data type associated with a digital entity to decide which display application should be invoked when the digital entity is retrieved and thus perform dynamic transformative migrations.

20. Data referenced by catalog query

Core capability definition
Mechanisms for attribute-based digital entity discovery

Functionality provided by the capability
Given the very large number of digital entities that are being archived, it is not possible to a priori know the logical names of all digital entities within a collection. A context is described for the digital entities by specifying semantic terms and associated attribute values. Discovery of a particular digital entity is then accomplished by querying on the attribute values. This requires that the user recognize the semantics inherent in the attribute names.

Example grid implementation
The mechanisms used to do discovery in data grids range from explicit creation of SQL commands, to specification of attribute names and values and automated generation of the required SQL. The latter approach requires the ability to characterize the table structure of the data grid metadata catalog, identify the foreign keys that are used between the tables, and generate the required joins. The result of the query generates either logical names, which can then be queried to discover the associated physical replicas. Alternatively, the metadata catalog query can result in direct access to the “nearest” copy of the associated physical file.

21. Containers for metadata

Core capability definition
Mechanisms for manipulating metadata attributes that are aggregated into a single file

Functionality provided by the capability
The use of XML annotated files makes it possible to assemble metadata attributes for either bulk metadata ingestion or bulk metadata export. By structuring the XML annotated file through application of either an XML DTD or XML Schema, a characterization of the collection can be created.

Example grid implementation
In data grids, metadata containers are primarily used for latency management. When metadata is extracted from a digital entity by application of an extraction template at the storage system, the attribute values are aggregated before transmission over the network. When databases are registered as objects within the logical name space, SQL command strings can be generated that extract metadata from the database. Again the metadata is aggregated into an XML file before it is moved over the network.

22. Distributed resilient scalable architecture

Core capability definition
Mechanisms to support fault tolerance and high access performance across distributed repositories

Functionality provided by the capability
The ability to scale requires automation of data management capabilities. Through use of a logical name space, storage repository abstractions, and information repository abstractions, it is possible to drive all data manipulation operations directly from an application. By designing the correct applications, any type of processing of a collection can be automated from metadata extraction, to encoding format transformation, to replication, etc. Resilience is accomplished through either replication for fault tolerance, replication for data assurance, or re-generation

through application of the deriving process. Both the resilience and automation mechanisms need to operate in a distributed environment, primarily because migration onto new technologies requires the ability to simultaneously access both the old and new forms of the technology.

Example grid implementation
Most data grids are implemented as federated client server architectures. Servers are installed at each storage system where data will be held. The servers map from the protocol of the local storage system to the storage repository abstraction. The servers can exchange data directly between themselves through third party transfer, making it possible to issue commands for replication that only involve the source and destination sites. Replication is used as the primary resiliency mechanism, with data accesses automatically failing over to a replica location if the desired copy was not available. Distribution is handled by federation of the servers, such that servers can communicate between each other independently of the driving client.

23. Specification of system availability

Core capability definition
Mechanisms to specify the permanency of the digital holdings, the access limitations, and the access availability

Functionality provided by the capability
An approach to data assurance is to move data from storage systems in which the guaranteed residency period is shorter than the desired period, to storage systems that can meet the assurance requirements. By putting the burden on reliability on the underlying storage systems, it is possible to force the storage systems to manage replicas for data assurance. This is typically done in hierarchical storage managers, which keep multiple copies of data on tape.

Example grid implementation
Most data grids rely on assurance specifications through the data grid metadata catalog, typically by adding attributes to characterize the type of storage repository as permanent tape storage, permanent disk cache under the control of the collection, temporary disk cache under the control of a system administrator, ephemeral disk cache that is subject to policy based purging. Copies of data are kept to assure permanence, with the copies geographically distributed to protect against disasters.

24. Standard error messages

Core capability definition
Mechanism to report error messages generated by all components of the persistent archive, from storage systems, to networking, to presentation errors.

Functionality provided by the capability
There are over one thousand error messages that can be generated across storage, network, and information repository environments. The organization of error messages into classes or severity is done to minimize the necessity of learning the meaning of each error message. Severity classes can include unrecoverable (must try another resource), recoverable (try again against the current resource), and advisory (non-fatal problem).

Example grid implementation
Data grids currently use their own standards for error messages. The data grid forum is examining the development of event based reliability systems that will generate a consensus on error message types. Actual implementations of error messages within data grids either report every error message back to the user, or classify the errors into a small set of classes.

25. Status checking

Core capability definition
Mechanism to report on the status of a request

Functionality provided by the capability
Status checking can be managed by event monitoring systems when single requests are made. When bulk processing is attempted, such as in metadata extraction and the movement of thousands of files, database technology is used to track the status of each individual component of a request.

Example grid implementation
Many of the data grid status checking mechanisms are done synchronously through notification on completion of a task. Some systems support dynamic status checking, such as the transmission of markers in the data flow to support transmission restart, or the creation of process flows where each processing stage is described by a characterization of the processing step. Status then corresponds to identifying which processing step was last completed. Resiliency in implemented by restarting from the last complete stage of the processing pipeline.

26. Authentication mechanism

Core capability definition
Mechanism to identify both individuals as single persons, and individuals as members of groups

Functionality provided by the capability
To manage the multiple access roles required for collection building, authentication mechanisms are needed to identify each person. In particular, curatorial and creation roles must be restricted to the archivists to ensure permanency of the archival holdings. For proprietary data, identification as members of groups is usually sufficient to manage access. Group based identification is appropriate where anonymity of access is required.

Example grid implementation
Data grids differentiate between the authentication systems used between administration domains, and the authentication systems used within an administration domain. The Grid Security Infrastructure uses Public Key Infrastructure and certificates to identify individuals. The certificates are managed by certificate authorities that follow specified managerial practices for the assignment of certificates to individuals. Alternatively, encrypted passwords and challenge response mechanisms are used to identify not only individuals, but also servers within the federated client server architecture. Local authentication systems are either Unix based, Kerberos based, or DCE based. The Generic Security Service API is used to map between the different authentication environments.

27. Specification of reliability against permanent data loss

Core capability definition
Mechanism to ensure survival of collection holdings across all types of failure mechanisms

Functionality provided by the capability
The assurance of reliability against permanent data loss has two components, protection of the original bits that comprise the digital entities, and protection of the mechanisms that identify the context used to organize the digital entities. Protection against data loss can be done through replication. Protection against information loss is much harder. The context can change over time through the addition of new material and the re-purposing of collections. The information content therefore requires both replication and snapshot mechanisms to ensure digital entities can be both identified and retrieved.

Example grid implementation
Hierarchical storage managers have traditionally safeguarded both the digital holdings and the metadata describing where the holdings are stored. The digital holdings are replicated. The metadata is replicated. Snapshots are taken of the metadata state at periodic intervals, and transaction logging is used to record all changes to the metadata. The transaction logs are replicated and periodically applied to the snapshots to guarantee that the state of the metadata catalog can be recreated. Similar approaches are needed in persistent archives to ensure reliability of the collection.

28. Specification of mechanism to validate integrity of data and metadata

Core capability definition
Mechanism to validate the authenticity of the digital holdings

Functionality provided by the capability
Authenticity requires showing that all operations that have been performed on a digital entity can be identified and characterized, that the metadata that is used to define the context for the digital entity is consistent with the operations that have been performed, and that the bits of the digital entity have not changed between transformative migrations. The consistency that can be maintained between the data and metadata is one of the primary advantages of the use of data grid technology with collection owned holdings.

Example grid implementation
Audit trails are used to record all accesses and operations that are performed on the digital holdings. Digital signatures and checksums are used to show that a digital entity has not been corrupted by disk, transmission, or recording errors. By checking the audit trails and comparing the recorded checksum with the current checksum, one can validate the integrity of the data. By examining the operations performed upon the digital entity, and the person who initiated the operation, one can show that only archivists have applied archival processes to the digital entities.

29. Specification of mechanism to assure integrity of data and metadata

Core capability definition
Mechanism to uniquely characterize a digital entity

Functionality provided by the capability
Through the use of the OAIS technology for specifying archival information packages (AIPs), it is possible to aggregate metadata and data into a single file. By signing or check-summing the AIP, one can determine that the content has not changed over time. By comparing the metadata within the AIP to that organized into the data grid catalog and applying the audit trail transformations, one can show that the digital entity corresponds to all recorded operations, and thus still contains the expected information and knowledge content.

Example grid implementation
The assurance of integrity is primarily managed in data grids by restricting operations on the digital holdings to the persons fulfilling the archival roles. The specification of a signature or checksum is inadequate if the signature can be forged through an unauthorized operation. Data grids get around this problem by working with collection owned data, and by auditing all operations done on a digital entity.

30. Virtual Data Grid

Core capability definition
Mechanism to create derived data products on demand

Functionality provided by the capability
The application of archival processes to digital entities can be characterized as a set of processing steps. The characterization can be stored as a process flow in the archive along with the digital entities, and organized in a logical name space sub-collection. A query against a collection for a digital entity can then be made against the collection attributes. If the query is not satisfied, a search can be done on the processing characterizations for the ability to generate the required derived data product. If the processing step is found, one then has to identify the required input files and input parameters. This requires knowledge about the relationships used to govern the archival process, and can be specified as part of the original query.

Example grid implementation
Virtual data grids use process characterizations based upon Directed Acyclic Graphs. These simple descriptions map output to input files for the multiple stages of a process pipeline. More sophisticated versions of process data flow are needed to incorporate knowledge about application of the processing steps; namely how to decide which digital entities are to be used as input to the processing stages. These process flow characterizations require the integration of concept spaces on top of the information catalogs managed by the data grids. The concept spaces specify the relationships that govern the application of the processing steps.

31. Knowledge repositories for managing collection properties

Core capability definition
Relationship management systems to describe the constraints used to form a collection of digital entities, or the properties of the resulting collection, or the organization of relationships within a digital ontology.

Functionality provided by the capability
The characterization, organization, and manipulation of relationships are managed by knowledge repositories. Given that multiple knowledge repositories can be created, a knowledge repository abstraction is needed to describe the set of operations that can be performed upon a concept space that is implemented within a knowledge repository.

Example grid implementation
The application of relationships that are organized in a knowledge repository requires the ability to generate logical inference rules or processing steps from the relationship description. Systems have been developed that generate logic rules for semantic relationships, spatial rules for manipulation of atlases, and procedural rules for applying processing steps. Each of these systems is typically implemented for a particular discipline. Generic mechanisms are needed for persistent archives. An example system is the mapping from RDF relationship syntax to Common Logic rules that can then be analyzed for collective properties.

32. Application of transformative migrations for encoding format

Core capability definition
Mechanism to migrate the encoding format to a new encoding standard

Functionality provided by the capability
The evolution of the encoding format of digital entities must be addressed by persistent archives along with the evolution of the supporting software and hardware infrastructure. A transformation of an encoding format to a new standard can be characterized as a processing step that is performed under persistent archive policy management control. The transformation would typically be applied when the collection holdings are migrated to a new media standard, as the entire collection must be read and processed. The transformative migrations can be applied to digital entities, digital ontologies, and even to encoding standards used to describe collections.

Example grid implementation

Grids provide the ability to execute procedures at the storage system where the digital entity resides, and to stream the digital entity through a sequence of filters. The procedures can be named and organized within the logical name space, and stored in the data grid along with the digital entities. This makes it possible to execute transformative migrations on digital entities as part of a media migration process.

33. Application of archival processes

Core capability definition
Mechanism to characterize and apply archival processes

Functionality provided by the capability
Many of the archival processes correspond to metadata extraction, collection formation, transformative migration, and data management. The abstractions needed to support these processes correspond to management of a logical name space, a storage repository abstraction for the operations that can be done on digital entities, and an information repository abstraction for the operations that can be done on catalogs in databases.

Example grid implementation
Data grids implement the abstraction levels needed to support the application of archival processes. The challenge is correctly characterizing the archival processes, and validating that the characterizations perform the desired functions.