

Data Grid Implementations

Reagan W. Moore (San Diego Supercomputer Center)
Scott Studham (Pacific Northwest National Laboratory)
Arcot Rajasekar (San Diego Supercomputer Center)
Chip Watson (Jefferson National Laboratory)
Heinz Stockinger and Peter Kunszt (CERN)

February 19, 2002

Data grids link distributed, heterogeneous storage resources into a coherent data management system. From a user perspective, the data grid provides a uniform name space across the underlying storage systems, while supporting retrieval and storage of files. In the high energy physics community, at least six data grids have been implemented for the storage and distribution of experimental data. Data grids are also being used to support projects as diverse as digital libraries (National Library of Medicine Visible Embryo project), federation of multiple astronomy sky surveys (NSF National Virtual Observatory project), and integration of distributed data sets (Long Term Ecological Reserve). Data grids also form the core interoperability mechanisms for creating persistent archives, in which data collections are migrated to new technologies over time. The ability to provide a uniform name space across multiple administration domains is becoming a critical component of national-scale, collaborative projects.

The Global Grid Forum is promoting the development of standards for the implementation of data grids. One of the challenges is defining the minimal set of functionalities that are needed to implement a data grid. Is it possible to define a common set of capabilities across all of the data grid, data collection, digital library, and persistent archive projects that are already in production? A second challenge is to understand the set of capabilities that may be required by future applications. To answer both questions, 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 National 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 PNNL 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), 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 the data grid architecture must support 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 or replica catalog, 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 table listing the capabilities is given in Appendix A. Terms used in the comparison are explained in Appendix B.

Common Data Grid Capabilities:

What is most striking is that common user requirements are emerging across all of the data grids. Appendix C 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.

Data Grid Operations

Given a consensus on the set of capabilities, a data handling system can then be characterized by the parameters needed to implement each capability. Grid operations can be defined that manage the exchange of the required parameters for initiating a particular capability. We have organized the capabilities into grid operations related to authentication, replication, replica catalog or logical name space management, reliable transport services, and subscription or collection interaction services. Note that it is possible to define a unique grid operation for each capability, and then aggregate grid operations to combine multiple capabilities into a consistent, robust, reliable grid service. In Appendix D, we list a set of core grid operations that can be created from the grid capabilities. Currently, the list contains grid operations from the SRB environment, the JASMine data grid, and the proposed interface for Storage Resource Managers. As additional grid operations are implemented, the proposed list will be expanded to include the new capabilities and input parameters.

A typical set of grid operations is listed below. Note that the grid operations can be implemented with just a subset of the capabilities provided by the data grids, and can be augmented with further functionality by integrating additional data grid capabilities.

1. Storage Resource authentication

- Support interoperability between data grids for authentication

2. Replication Service

- Coordinate replica creation with replica catalog metadata update
- Could guarantee that the replica is created correctly (fault tolerance)
- Could support replication across k of n storage sites
- Could support out of band registration, bulk file registration, bulk export of attributes
- Could support authentication of replicas using a UUID and a hash/checksum

3. Replica Catalog (Logical Name Space Management)

- Support organization of logical names in collections / folders / directories
 - Support operations on metadata in logical name space
 - Support information discovery
- 4. Reliable Transport service**
- Support operations on data
 - Support partial file transfer
 - Support interface to SRM for asynchronous access
 - Could support physical organization of files
 - Could support digital entity registration
 - Could support fault tolerance mechanisms
 - Could support latency management functions at the remote site
 - Could support transfer of control sequences for latency management
 - Could support bulk transfer of metadata
 - Could support access to databases
- 5. Subscription Service**
- Support catalog update automation, such as catalog replication, catalog synchronization, and catalog mirroring
 - Could support user subscribed delivery of newly registered digital entities into a private user catalog
 - Could support checking of authenticity of catalog

The underlying grid operations for each category are explained in detail in Appendix D. Common grid operations are listed, followed by extended grid operations that take advantage of the extended capabilities listed in Appendix A. For each grid operation, a set of required input parameters is proposed that would be used to implement the grid operation.

One of the goals is to define a common set of input parameters for each grid operation. A general definition of the semantics to associate with each input parameter is also needed for it to become possible to implement services based on current web technology (WSDL and SOAP). To initiate a dialog, the semantic meaning of the input parameters for the Storage Resource Broker services is listed at the end of Appendix D.

Conclusion:

By comparing the implementations of seven different data grids, a common set of data management capabilities have been identified for accessing data in distributed environments. Based upon the common capabilities, a set of core operations have been proposed for data discovery, data access, and data movement. Extended operations have also been proposed that provide additional logical name space manipulation, latency management, and data management. For each operation, a set of parameters has been listed, along with the semantic meaning of each parameter.

The proposed set of core and extended operations are open to debate. One approach to resolve which operations are appropriate, is to compare implementations of Web services based upon the WSDL and SOAP protocols. If a consensus is reached on the parameters

associated with each WSDL service, the data grid community will then be able to provide a uniform service interface to the existing data grids.

Acknowledgements:

The data grid and toolkit characterizations were provided by the following persons. This comparison was only possible through their support. Igor Terekhov (Fermi National Accelerator Laboratory), Torre Wenaus (Brookhaven National Laboratory), Scott Studham (Pacific Northwest National Laboratory), Chip Watson (Jefferson Laboratory), Heinz Stockinger and Peter Kunszt (CERN), Ann Chervenak (Information Sciences Institute, University of Southern California).

Appendix A. A comparison of the capabilities provided by data grids

In the following table, areas where an implementation is planned for a capability are marked with “P”. Areas where information has not been received are left blank. The capabilities have been organized into eleven categories, with up to twenty different capabilities per category. A total of 152 capabilities are characterized. Over three-quarters of the capabilities (120) have been implemented in at least two of the data grids. About one-third of the capabilities (50) have been implemented in at least five of the data grids. These 50 capabilities comprise the current core features of data grids.

Capability	EDG	Globus Toolkit	JAS-Mine	Magda	SAM	SDM	NASA	SRB
Logical name space	Yes	Yes	Yes	Yes	Yes	Yes		Yes
Logical name space independence from physical name space	Yes	P	Yes	Yes	Yes	Yes		Yes
Hierarchical logical folders	P	P	Yes	No	No	Yes		Yes
Unix node operations on logical name space – directory manipulations to rename, delete, create and remove files and folders	No	No	Yes	Yes	Yes	No		Yes
Recursive operations on logical name space directories for both store and retrieval	No	No	Yes	No	No	No		Yes
Deletion of entities from logical name space	Yes	Yes	Yes	Yes	Yes	Yes		Yes
Delete by setting a deletion attribute	No	No	No	No	Yes	Yes		Yes
Delete by removing attribute	Yes	Yes	Yes	Yes	Yes	Yes		Yes
Soft links between objects in logical folders so that a single file can be listed in multiple folders	P	P	P	Yes	Yes	Yes		Yes
Data referenced by catalog owned by a user ID	Yes	Yes	Yes	Yes	Yes	Yes		Yes
Data referenced by catalog owned by a Collection ID	P	P	No	Yes	Yes	Yes		Yes
Registration of files as objects in logical name space	Yes	Yes	Yes	Yes	Yes	Yes		Yes
Registration of databases as objects in logical name space	No		No	No	Yes	No		Yes
Registration of database blobs as objects in logical name space	No	No	No	No	Yes	No		Yes
Registration of persistent handles as objects in logical name space	No		No	No	No	No		Yes

Capability	EDG	Globus Toolkit	JAS-Mine	Magda	SAM	SDM	NASA	SRB
Logical name space Attributes	Yes	Yes	Yes	Yes	Yes	Yes		Yes
System level attributes	Yes	Yes	Yes	Yes	Yes	Yes		Yes
Replica attributes for storage location, local file name	Yes	Yes	Yes	Yes	Yes	Yes		Yes
Replica attributes for type of storage system	No	No	No	Yes	Yes	Yes		Yes
Role based access control lists for data in logical name space	P		P	P	No	Yes		Yes
Group access control lists for data in logical name space	P		P	P	Yes	Yes		Yes
Access control lists for logical name space attributes, to control who can see, add, and change metadata	P	No	No	No	Yes	Yes		Yes
Extended set of user roles (administrator, collection curator, annotator, user)	P		No	Yes	Yes	Yes		Yes
Access control lists for resources	Yes	Yes	No	No	Yes	No		Yes
I/O access pattern	No		Yes	No		No		Yes
Version attribute	No	No	Yes	Yes	Yes	No		Yes
Audit trails for updates and/or accesses	No	No	Yes	No	Yes	Yes		Yes
User level attributes	Yes	No	Yes	Yes	Yes	Yes		Yes
User defined attributes for data entities	Yes	No	Yes	Yes	Yes	Yes		Yes
User defined attributes for collections	P		No	No	Yes			Yes
Annotation attributes	Yes	No	No	Yes		Yes		Yes
User profiles to describe storage usage history	No		No	No		No		P
Discipline specific attributes	Yes	Yes	Yes	Yes	Yes	Yes		Yes
Dublin Core attributes	No	No	No	No		No		Yes
Template based metadata extraction for catalog attributes	No		No	P				Yes
External catalog accessible for additional attributes	No	Yes	No	P		No		Yes
Physics tags	P		No	No	Yes			Yes

Capability	EDG	Globus Toolkit	JAS-Mine	Magda	SAM	SDM	NASA	SRB
Attribute manipulation	Yes	Yes	Yes	Yes	Yes	Yes		Yes
Query interface to discover files by attributes	P			Yes	Yes	Yes		Yes
Automated attribute generation for size, time stamp	Yes	Yes	Yes	Yes	Yes	Yes		Yes
User managed synchronous attribute update	Yes	Yes	Yes	Yes	Yes	Yes		Yes
Asynchronous annotation of objects in logical name space	Yes	No	Yes	Yes	Yes	Yes		Yes
Export of attributes as XML file or python file	Yes		No	No	Yes			Yes
Bulk asynchronous load of attributes	Yes	No	No	Yes	Yes	Yes		Yes
Bulk asynchronous load of attributes from XML or python file	No	No	No	No	Yes	No		Yes

Capability	EDG	Globus Toolkit	JAS-Mine	Magda	SAM	SDM	NASA	SRB
Data manipulation	Yes	Yes	Yes	Yes	Yes	Yes		Yes
Synchronous creation of replicas with associated metadata creation	Yes	Yes	Yes	Yes	Yes	No		Yes
Asynchronous creation of replicas	P	P	Yes	Yes	Yes	Yes		Yes
Registration of user owned data as a replica of an existing object in the logical name space	No	No	No	No	No	No		Yes
Master instances of replicas	P		Yes	Yes				Yes
Load balancing across physical resources	No		Yes	No	Yes	Yes		Yes
Containers for aggregating small files	No	No	No	No	No	No		Yes
Logical aggregation of objects for storage on tape	No	No	Yes	No	Yes	No		No
Container locking on writes	No	No	No	No	N/A	No		Yes
Updates to containers by appending data	No	No	No	No		No		Yes
Replication of containers	No	No	No	No		No		Yes
Container replica invalidation of updates	No	No	No	No		No		Yes
Staging of containers from archive to disk	No	No	No	No		No		Yes
Synchronization of containers to archives	No	No	No	No		No		Yes
Multiple disk caches for containers	No	No	No	No		No		Yes
Multiple archives for storing containers	No	No	No	No		No		Yes
Logical container names that represent multiple physical containers	No	No	No	No		No		Yes

Capability	EDG	Globus Toolkit	JAS-Mine	Magda	SAM	SDM	NASA	SRB
Data Access	Yes	Yes	Yes	Yes	Yes	Yes		Yes
Parallel I/O support	Yes	Yes	Yes	Yes	Yes	No		Yes
Parallel I/O on get/put commands	Yes	Yes	Yes	Yes	Yes	No		Yes
Parallel I/O on partial file reads/writes	Yes	Yes	No	No	No	No		No
Transmission status checking at the file level	Yes	Yes	Yes	Yes	Yes	No		Yes
Transmission block tagging to support transmission restart after interruption	Yes	Yes	Yes	No	No	Yes		No
Transmission restart after interruption at application level	Yes	Yes	Yes	Yes	Yes	Yes		Yes
Storage completion at end of single write	Yes	Yes	Yes	Yes	Yes	Yes		Yes
Replication completion when write to k of n physical resources	No	No	No			No		Yes
Standard error messages from storage systems, network, and data handling system	Yes	Yes	Yes	Yes	Yes	Yes		Yes
Striping support	Yes	Yes	Yes	No	No	Yes		P
Thread safe client	Yes		Yes	No	Yes	Yes		Yes

Static network tuning	Yes	Yes	Yes	No	P	No		Yes
Dynamic network tuning (window and buffer size)	No	Yes	No	No	P	No		No
GridFtp protocol support	No	Yes	P	Yes	P	No		P
TCP/IP custom control protocol	No	No	No	No	P	No		Yes
Java parallel custom control protocol	No	No	Yes	No	No	No		No
User-selectable transfer (Gridftp, scp, ..)	Yes		Yes	Yes				No
Push and Pull data movement			Yes	Yes				Yes

Capability	EDG	Globus Toolkit	JAS-Mine	Magda	SAM	SDM	NASA	SRB
Multiple Access APIs	Yes	Yes	Yes	Yes	Yes	Yes		Yes
Remote data access by I/O redirection from Linux or Solaris system I/O calls	Yes	No	No	No		No		Yes
C I/O library API	Yes	Yes	No	No		Yes		Yes
C++ I/O library API	Yes	No	No	Yes	Yes	Yes		Yes
Command line interface	Yes	Yes	Yes	Yes	Yes	Yes		Yes
Java interface	Yes	P	Yes	Yes		Yes		Yes
Web interface	P	No	Yes	Yes	Yes	No		Yes
Web Services Interface (WSDL)	P	P	Yes					P
Visual Basic interface	No	No	No	No	No	Yes		Yes
XML query interface	P		Yes	No	No	P		Yes
DLL/API interface for Python	P		No	No		No		Yes
Predicate assertion interface	No		No	No		No		Yes
SDLIP interface	No		No	No		No		P
Windows browser interface	No	No	No	No		Yes		Yes

Capability	EDG	Globus Toolkit	JAS-Mine	Magda	SAM	SDM	NASA	SRB
Distributed client-server architecture	Yes	Yes	Yes	Yes	Yes	Yes		Yes
Federated client server	Yes	Yes	Yes	Yes		Yes		Yes
Distributed servers	Yes	Yes	Yes	Yes	Yes	Yes		Yes
Distributed storage systems	Yes	Yes	Yes	Yes	Yes	Yes		Yes
64-bit storage space	Yes							Yes
Logical resources that represent multiple physical resources	No	No	No	Yes	Yes	Yes		Yes
Third party transfer from storage system to specified remote destination	Yes	Yes	Yes	No	No	Yes		Yes
GSI authentication	Yes	Yes	No	Via gsiftp	P	P		Yes
PKI authentication	Yes	Yes	Yes	No	P	No		Yes
Challenge response authentication	Yes		No	No	Yes	No		Yes
Ticket based access control for number of accesses and restricted time period for access	No		No	No	Partial	P		Yes

Capability	EDG	Globus Toolkit	JAS-Mine	Magda	SAM	SDM	NASA	SRB
Latency Management	Yes	Yes	Yes	Yes	Yes	Yes		Yes
Streaming	Yes	Yes	Yes	Yes		Yes		Yes
Caching	Yes	Yes	Yes	Yes	Yes	Yes		Yes
Prefetch (partial file access)	Yes	Yes	No	No		No		Yes
Containers for data			No	No	Yes	No		Yes
Containers for metadata			No	No	Yes	No		Yes
Remote I/O proxies for aggregating I/O commands, remote data filtering, metadata extraction	No	Yes	No	No		No		Yes
Remote Proxies through DataCutter	No	No	No	No		No		Yes
Remote Proxies through GridFTP	No	Yes	No	No		No		No
Staging	Yes		Yes	Yes	Yes	Yes		Yes
Status checking	Yes		Yes	Yes	Yes	No		Yes
Replication	Yes	Yes	Yes	Yes	Yes	No		Yes

Capability	EDG	Globus Toolkit	JAS-Mine	Magda	SAM	SDM	NAS A	SRB
Multiple system support	Yes	Yes	Yes	Yes	Yes	Yes		Yes
Storage Resource Manager interface	Yes	No	Yes	P	Yes	No		Yes
Database interface for reading/writing blobs	No	No	No	No		No		Yes
Database interface for queries to relational database registered as a data object	Yes	No	No	No		No		Yes
Database interface for exporting attributes as an XML file	Yes	No	No	No		No		Yes
Archive interface to at least one archive	Yes	Yes	Yes	Yes	Yes	Yes		Yes
Archive interface to HPSS	Yes	Yes	No	Yes		No		Yes
Archive interface to DMF	Yes	Yes	No	No		No		Yes
Archive interface to ADSM	Yes	No	No	No		No		Yes
Archive interface to Enstore	No			No	Yes			No
Archive interface to UniTree	Yes	No	No	No		No		Yes
Archive interface to JASMine	No	No	Yes	No		No		No
Archive interface to HPSS for storing file sizes greater than 2 GB	Yes	No	No	No		No		Yes
Archive interface to Castor	Yes		No	Yes				No
Single catalog server, database technology used to replicate catalog	Yes	Yes	Yes	Yes		Yes		Yes
Hierarchical distributed catalog	P	P	No	No	No	No		No

Capability	EDG	Globus Toolkit	JAS-Mine	Magda	SAM	SDM	NAS A	SRB
Performance enhancements	Yes	Yes	Yes	Yes	Yes	Yes		Yes
Performance for import/export of files greater than 20 files/sec	Yes	Yes	Yes	Yes		Yes		Yes
Performance for import/export of files greater than 1100	Yes	No	No	Yes		No		Yes

files/sec								
Bulk metadata load	P		No	Yes	Yes	No		Yes
Pre-spawned processes for data transfers	No		No	No		No		Yes
Database index optimization	Yes		Yes	No		Yes		Yes
Database communication tuning	Yes		Yes	No		No		Yes

Capability	EDG	Globus Toolkit	JAS-Mine	Magda	SAM	SDM	NAS A	SRB
Robustness: Fault tolerance and error handling	Yes		Yes	Yes	Yes			Yes
Automatic fail over to alternate replica when the first copy is unavailable	No		No	Yes	Yes	No		Yes
Automatic retrials in metadata catalogue access	No			No	Yes			No
Automatic retrials in archive access	No		Yes	No	Yes			P
Data transfer resumption upon system restart (crash and reboot), transparently to the user jobs	Yes		Yes	No	Yes			P
Single-command resumption of very long user jobs upon system restart	Yes		P	Yes	Yes			No
Configurable time-outs on user usages of every resource (to protect against abandoned/misbehaved user jobs)	P		Yes	P	Yes			No
Handling of mass storage system software errors, including protocol non-compliance and other aberrations	No		Yes	Yes	Yes			Yes
Handling of underlying file transfer tools (such as ftp) errors, including non-protocol compliance	Yes		P	Yes	Yes			Yes
File checksum	Yes	P	Yes	No	No	No		Yes
Number of capabilities (present or planned) (total of 152)	98	60	80	76	85	64		136

Appendix B. Definition of terms used in the capability comparison

Terms used in Appendix A to describe data grid capabilities are defined below.

- Registration corresponds to adding objects to the logical name space, creating a logical name and storing a pointer to the file name used on the storage system
- Attributes represent information that is managed for each object that is registered into the logical name space
- Folders are equivalent to directories in a file system, but are used to organize objects in the logical name space
- Soft links represent the cross registration of a single physical data object into multiple folders 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 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 file systems.
- A container is an aggregation of multiple data files into a single file
- Curation control corresponds to the administration tasks associated with creating and managing a logical collection
- 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.
- 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.
- Load balancing for a logical name space consists of distributing digital objects across multiple storage systems
- 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 completion at end of single write corresponds to synchronous data writes
- 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
- Federated server architecture refers to the ability of distributed servers to talk among themselves without having to communicate through the initiating client.
- 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
- Bulk metadata load is the ability to import attribute values for multiple objects registered within the logical name space from a single input file.
- 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

- DataCutter is the data filtering service developed by Joel Saltz at the Ohio State University, which is executed directly on a remote storage system.

Appendix C. 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 data grids that were surveyed. 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 given 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 D: Data grid operations and associated parameters

The proposed grid operations can be decomposed into the simple actions that may be initiated by a user. For each major category of grid service, a set of simple grid operations is defined, along with the input and output parameters that would be used by the grid operation. The grid operations are organized into core standard grid operations, and extended grid operations. The grid operations are based upon implementations that are being created for the Storage Resource Manager and the Storage Resource Broker. As additional implementations are developed, the list of operations is expected to evolve with new semantics and input parameters.

Grid operations can be defined for multiple levels of the grid infrastructure. We will separate the operations into capabilities for:

- Replica Catalog – manages mapping of a global logical name space to the logical name space of the SRM. Also supports manipulations done on the logical name space managed by the replica catalog. These operations correspond to manipulation of the directory structure of the replica catalog, and the creation of a replica.
- Storage Resource Managers – manipulations done on the name space supported by the SRM. These operations are proposed as the standard operations for storage systems that are accessed by a grid. SRMs support staging of files, asynchronous access, and management of files on local resources through a local logical name space.
- Reliable Transfer – the operations that involve movement of files between SRMs.

While twenty-one grid operations are proposed as core grid operations within the data grid, they can be applied either as operations within an SRM, or as operations on the digital entities registered into a Replica Catalog. Thus many of the operations are listed twice, once for each name space, and are highlighted in bold in the following table. A set of 15 grid operations is defined for asynchronous interactions with Storage Resource Managers.

One of the extensions to the Unix file system semantics is the concept of containers for the physical aggregation of files before storage into an archive. In the core grid operations where containers may be referenced, the associated container input parameter is italicized to indicate that it is an extension.

At the end of the table, all of the parameters are listed with an explanation of the semantic meaning of each parameter. Using these definitions, it is possible to create a WSDL specification for each grid operation. Note that many of the commands can be used to manipulate either the logical name space collection/sub-collection hierarchy, specified by a “C” in the command, or the digital objects registered into the logical name space, specified by a “D” in the command. For each type of data management system, the logical name space that is referenced is defined. In a few situations, such as copying a file from a local file system into a replica catalog, or writing into the local file system, a physical file name is required.

D.1 Replica Manager Interactions

The name space used by the Replica Manager is a logical name space that is independent of the name space used within an SRM. The Replica Manager manages the mapping from the replica manager logical name space to the SRM name space. The Replica Manager can organize the logical name space as a collection/sub-collection hierarchy, or as a directory/subdirectory hierarchy. In the collection/sub-collection hierarchy, the set of attributes associated with each sub-collection can be varied, making it possible to integrate multiple collections into a single name space.

D.1.A. Replica Manager authentication

Both the SRM and the Replica Manager need to be able to authenticate users, and manage access control lists for both directories and digital entities within their name spaces.

1. **loginUser** sename, userDomain, Password, srbHost, srbPort, sessionLength → returns sessionId
 Uses encrypted password or X.509 certificate to authenticate a user. Each user is represented by the triplet of (sename, userDomain, Password), or (username, group, Password). A sessionId is issued to permit multiple accesses over a specified time period without requiring re-authentication. The user specifies which data handing system is being accessed by stating the host (srbHost) and the port number on the host (srbPort).
2. **logoutUser** sessionId
 Terminates a session. A session is automatically terminated after the duration specified by the “session length”.
- Password specific authentication commands
 1. *registerUser* sename, userAddr, userEmail, userPhone, srbHost, srbPort
 A user can self register into the authentication environment if they access the system from inside a firewall and have a valid E-mail account.
 2. *changePassword* sename, userDomain, Password, newPassword, srbHost, srbPort
 Supports user specified password changes.
 3. *requestPassword* sename, userDomain, srbHost, srbPort
 A user can ask for their password to be re-sent by an administrator.
- Extended authentication mechanisms for ticket based access
 1. *issueTicket* sessionId, collName, dataName, sename, userDomain, beginTime, endTime, numberOfAccesses, recursiveFlag
 A user can request a ticket (“string”) that can be used for read only access to the collection. The ticket has an associated time period for use, and a maximum number of accesses. The ticket can be restricted to an individual data set, or to a collection, or extended to all sub-collections within a collection.
 2. *loginWithTicket* ticket → returns sessionId

- Access the collection using a ticket
3. *showTicketInfo* sessionId, ticket
List the access attributes associated with a ticket
 4. *showCDInfoUsingTicket* sessionId, collName, dataName
List attributes about either a collection or dataset using a ticket for authentication. If the “data name” is left blank, attributes are listed about the collection.
 5. *showDataUsingTicket* sessionId, collName, dataName
Display a data set using a ticket for authentication by using a helper application derived from the “dataType” attribute.
 6. *downloadDataUsingTicket* sessionId, collName, dataName
Copy a data set to the local file system using a ticket for authentication.

D.1.B Logical Name Space Management within Replica Catalog

The file references are to the logical name space used by the Replica catalog to manage data. References to physical files are interpreted as references to the files managed by an SRM, or to files in an unmanaged storage system.

- Support organization of logical names in collections / folders / directories
 1. **createCollection** sessionId, parentCollName, newCollName, containerName, METADATALIST
Create a sub-collection under a parent collection, specify whether a container is used to aggregate data files stored in the sub-collection, and provide the metadata attributes used to characterize the sub-collection.
 2. **deleteC** sessionId, collName
Delete a collection.
 3. **linkC** sessionId, collName, targetCollectionName
Use soft links to register a sub-collection into the “target collection name”.
- Extended logical name space services
 1. *moveC* sessionId, collName, targetCollName, resourceName, containerName
Move a collection into a “target collection name”.
 2. *showCInfo* sessionId, collName
List the attributes associated with a collection
 3. *DirectoryListing*
List a directory within a physical file system that has been registered into the logical name space through a shadow link.
 4. *accessControlC* sessionId, collName, userName, domainName, accessPermission
Set the access control for a logical collection for the specified (user name, domain name). Access controls can be set for owner, co-owner, write, annotate, read, null.
 5. *insertUpdateComments* sessionId, collName, dataName, newComments
Owner-created comments can be added to a “data name” within a collection.
- Support operations on metadata in logical name space

1. **insertMyMeta** *sessionId, collName, METADATALIST*
Add user defined metadata to a collection consisting of a triplet (attribute name, attribute value, and units).
 2. **deleteMyMeta** *sessionId, collName, metaDataId*
Delete user defined metadata. The metadata ID is returned in the “show my metadata” command.
- Extended operations on metadata in logical name space
 1. *copyMyMeta* *sessionId, collName, dataName, targetCollName, targetDataName*
Copy user defined metadata from one “data object” to a target data object.
 2. *modifyMyMeta* *sessionId, collName, dataName, METADATALIST-with-ID*
Modify user defined metadata, specifying the metadata ID for each modified attribute.
 3. *insertAnnotation* *sessionId, collName, dataName, ANNOTATIONLIST*
Add a user-defined annotation to a data object. The annotation includes a time stamp and the identity of the person making the annotation.
 4. *deleteAnnotation* *sessionId, collName, dataName, annotationId*
Delete a specific annotation. The “annotation ID” is returned by the “show my Metadata” command.
 5. *copyAnnotation* *sessionId, collName, dataName, targetCollName, targetDataName*
Copy an annotation from one “data object” to a target data object.
 6. *modifyAnnotation* *sessionId, collName, dataName, ANNOTATIONLIST-with-ID*
Modify an annotation specified by an “annotation ID”.
 - Support information discovery
 1. **queryInfo** *sessionId, QUERYCONDITIONLIST*
Issue a query comprised of a set of (attribute, query-condition, attribute-value) triplets. Valid query conditions include (=,<,>,like)
 2. **showMyMeta** *sessionId, collName, dataName*
List the metadata for the specific “data object”.
 3. **queryMyMeta** *sessionId, QUERYCONDITIONLIST*
Issue a query against the user defined metadata specifying a triplet of (attribute name, query condition, attribute value).
 - Extended services for information
 1. *showUserInfo* *sessionId*
List the attributes that characterize you (address, domain, E-mail, telephone, user name).
 2. *showResourceInfo* *sessionId*
List the attributes that characterize a storage resource.
 3. *showContainerInfo* *sessionId*
List all of the containers belonging to you.

D.1.C Replication Creation

- Coordinate replica creation with replica catalog metadata update
 1. **replicateData** sessionId, collName, dataName, resourceName
Create a replica of a digital object onto the resource specified by resourceName. Note that the “data name” includes both the logical name and the replica number.
 2. **removeData** sessionId, collName, dataName
Delete a replica by specifying the “data name” which includes both the logical name and the replica number.
- Could guarantee that the replica is created correctly (fault tolerance)
- Could support replication across k of n storage sites
- Could support out of band registration, bulk file registration, bulk export of attributes
- Could support authentication of replicas using a UUID and a hash/checksum

D.2 Interactions with a Storage Resource Manager

A Storage Resource Manager implements a logical name space to integrate access across local resources. The SRM can support staging of files, management of replicas between local disk and tape systems, and control of interactions with the local resources. Two versions of the SRM interface are discussed; one oriented towards synchronous manipulation of files within the SRM, and one oriented towards asynchronous access towards the SRM.

D.2.A. Storage Resource Manager authentication

- Support interoperability between data grids for authentication
 1. **loginUser** userName, userDomain, Password, srbHost, srbPort, sessionLength --> returns sessionId
Uses encrypted password or X.509 certificate to authenticate a user. Each user is represented by the triplet of (userName, userDomain, Password), or (username, group, Password). A sessionId is issued to permit multiple accesses over a specified time period without requiring re-authentication. The user specifies which data handing system is being accessed by stating the host (srbHost) and the port number on the host (srbPort).
 2. **logoutUser** sessionId
Terminates a session. A session is automatically terminated after the duration specified by the “session length”.

D.2.B. Storage Resource Manager Logical Name Space Management

- Support organization of logical names in collections / folders / directories
 1. **createCollection** sessionId, parentCollName, newCollName, containerName, METADATALIST
Create a sub-collection under a parent collection, specify whether a container is used to aggregate data files stored in the sub-collection, and provide the metadata attributes used to characterize the sub-collection.
 4. **deleteC** sessionId, collName

- Delete a collection.
- 5. **linkC** sessionId, collName, targetCollectionName
Use soft links to register a sub-collection into the “target collection name”.
- Extended logical name space services
 - 2. *moveC* sessionId, collName, targetCollName,
resourceName, containerName
Move a collection into a “target collection name”.
 - 4. *showCInfo* sessionId, collName
List the attributes associated with a collection
 - 5. *DirectoryListing*
List a directory within a physical file system that has been registered into the logical name space through a shadow link.
 - 6. *accessControlC* sessionId, collName, sername,
domainName, accessPermission
Set the access control for a logical collection for the specified (user name, domain name). Access controls can be set for owner, co-owner, write, annotate, read, null.
 - 7. *insertUpdateComments* sessionId, collName, dataName, newComments
Owner-created comments can be added to a “data name” within a collection.
- Support operations on metadata in logical name space
 - 3. **insertMyMeta** sessionId, collName, METADATALIST
Add user defined metadata to a collection consisting of a triplet (attribute name, attribute value, and units).
 - 4. **deleteMyMeta** sessionId, collName, metaDataId
Delete user defined metadata. The metadata ID is returned in the “show my metadata” command.
- Extended operations on metadata in logical name space
 - 7. *copyMyMeta* sessionId, collName, dataName, targetCollName,
targetDataName
Copy user defined metadata from one “data object” to a target data object.
 - 8. *modifyMyMeta* sessionId, collName, dataName,
METADATALIST-with-ID
Modify user defined metadata, specifying the metadata ID for each modified attribute.
 - 9. *insertAnnotation* sessionId, collName, dataName,
ANNOTATIONLIST
Add a user-defined annotation to a data object. The annotation includes a time stamp and the identity of the person making the annotation.
 - 10. *deleteAnnotation* sessionId, collName, dataName, annotationId
Delete a specific annotation. The “annotation ID” is returned by the “show my Metadata” command.
 - 11. *copyAnnotation* sessionId, collName, dataName, targetCollName,
targetDataName
Copy an annotation from one “data object” to a target data object.
 - 12. *modifyAnnotation* sessionId, collName, dataName,

ANNOTATIONLIST-with-ID

Modify an annotation specified by an “annotation ID”.

- Support information discovery
 1. **queryInfo** sessionId, QUERYCONDITIONLIST
Issue a query comprised of a set of (attribute, query-condition, attribute-value) triplets. Valid query conditions include (=,<,>,like)
 2. **showMyMeta** sessionId, collName, dataName
List the metadata for the specific “data object”.
 3. **queryMyMeta** sessionId, QUERYCONDITIONLIST
Issue a query against the user defined metadata specifying a triplet of (attribute name, query condition, attribute value).
- Extended services for information
 1. *showUserInfo* sessionId
List the attributes that characterize you (address, domain, E-mail, telephone, user name).
 2. *showResourceInfo* sessionId
List the attributes that characterize a storage resource.
 3. *showContainerInfo* sessionId
List all of the containers belonging to you.

D.2.C. Storage Resource Manager commands for asynchronous access. Note that the SRM commands assume that the file references are relative to the SRM Site URLs. The SRM in turn maps from the Site URL to a physical file name, and can return a handle (Transfer URL) indicating the address of a file that is ready for transport.

1. **Get** SURLS, Protocols
Request the SRM to reserve disk space, migrate a file to disk, and pin the file. The file is specified by a site specific URL, along with the data transfer protocol. A Request_ID is returned along with the request status.
2. **Put** Src_file_names, dest_file_names, sizes, wantPermanent, Protocols
Reserve disk space for the specified destination file name, and copy the source file onto the disk, using the specified data transfer protocol.
4. **List** SURL
Equivalent to Unix “ls” command
5. **mkdir** SURL
Equivalent to Unix “mkdir” command
6. **MkPermanent** SURLS
Mark a file named by the site specific URL, as permanent on the disk.
7. **Pin** TURLs
Keep a file named by the Transfer URL (the copy on disk) resident on disk for a default time period.
8. **UnPin** TURLs, Request_ID
Release a file that was copied to the disk under the Transfer URL name.
9. **getRequestStatus** Request_ID
Return the status of the named request including size, state, FileId, transfer URL, owner, group, permissions, checksum, checksum type, pin status,

permanence, estimated time for availability on disk, source file name, destination file name, order in the queue, and cache state.

10. **getFileMetaData** URLs
Return the attributes for the site specific URL, including the size, owner, group, permissions, checksum, checksum type, pin state, permanence, cache state.
11. **getProtocols** -
Return a list of SRM supported protocols
12. **getEstGetTime** URLs, Protocols
Return the status of the files named by the site specific URLs
13. **getEstPutTime** Src_file_names, dest_file_names, sizes, wantPermanent, Protocols
Return the status of the files named by the site specific URLs
14. **setFileStatus** Request_ID, fileId, State
Update a file from ready to running to initiate the data transfer.
15. **AdvisoryDelete** URLs
Recommend that the file named by the site specific URL be deleted from disk first.

D.3 Reliable Transport service

- Support operations on data
 1. **downloadData** sessionId, collName, dataName
Copy a data object into the local file system
 2. **removeData** sessionId, collName, dataName
Delete a data object from the logical name space and from storage.
 3. **renameData** sessionId, collName, dataName, newdataName
Rename a data object in the logical name space.
 4. **copyData** sessionId, collName, dataName, targetDataName, targetCollName, resourceName, *containerName*
Copy a data object from the specified (logical collection name, data name) to a target (logical collection name, data name).
 5. **moveD** sessionId, collName, dataName, targetDataName, targetCollName, resourceName, *containerName*
Move a data object within the logical name space to a new collection, and copy the data onto the specified resource.
 6. **accessControlD** sessionId, collName, dataName, userName, domainName, accessPermission
Set the access permissions for a specific data object within the logical name space for the specified user. Access controls can be set for owner, co-owner, write, annotate, read, null.
 7. **showDInfo** sessionId, collName, dataName
List the attributes for the specified logical data object name.
 8. **linkD** sessionId, collName, dataName, targetDataName, targetCollName
Create a soft link for the logical data name into the target collection.
 9. **ingestFile** sessionId, collName, newdataName,

resourceName, *containerName*, dataType,
inputFileBuffer, METADATALIST

Add a file name to the logical name space and copy the data onto the specified storage resource.

- Extended operations on data
 1. *reingestFile* sessionId, collName, dataName, inputFileBuffer
Reload a data set from the local file system into the logical name space and associated storage resource.
 2. *registerFile* sessionId, collName, newdataName, resourceName, filePath, dataType, METADATALIST
Create a shadow link to a file in a physical file system. This registers the file into the logical name space, without making a copy of the file.
 3. *showData* sessionId, collName, dataName
Invoke a helper application defined by the attribute “dataType” and display a data object in the logical name space.
 4. *deleteD* sessionId, collName, dataName
Delete a data set. This is the same operation as “removeData”.
 5. *lockData* sessionId, collName, dataName, lockType
Create a lock on the specified logical file name. The types of locks correspond to the types of access control (write, annotate, read, null)
 6. *checkoutData* sessionId, collName, dataName
Create a write lock on a logical data object, and restrict writes to this sessionId.
 7. *checkinData* sessionId, collName, dataName, inputFileBuffer
Reload the logical data object and release the write lock.
- Support partial file transfer
- Extended support for physical organization of files
 1. *createContainer* sessionId, containerName, containerCollection, Size, dataType, resourceName
Specify a logical container that will be used to aggregate files before storage into an archive specified by the resource name.
 2. *dropContainer* sessionId, containerName
Delete a container.
- Extended support for digital entity registration
 1. *registerDirectory* sessionId, collName, newDirectoryName, resourceName, directoryPath, METADATALIST
Register a directory in a physical file system as an object in the logical name space. This effectively creates a shadow link to the directory.
 2. *registerURL* sessionId, collName, newdataName, URL, METADATALIST
Register a URL as an object in the logical name space.
 3. *registerSQL* sessionId, collName, newdataName, resourceName, SQL, METADATALIST
Register a SQL command sequence as an object in the logical name space.

METADATALIST-with-ID	stands for the list of information needed when inserting metadata into an existing attribute
newCollName	new name for a collection/sub-collection
newComments	new comments that will be associated with a digital entity
newdataName	new name to be used for a digital entity
newDirectoryName	new logical collection/sub-collection name to associate with a physical directory name
newPassword	new user password
numberOfAccesses	maximum number of accesses for which a ticket may be used
parentCollName	collection or sub-collection name that will be used as the parent collection
Password	user password
QUERYCONDITIONLIST	stands for the list of conditions used to process a Query. Allowed operations include (=, <, >, like)
recursiveFlag	ticket access control for subdirectories
resourceName	logical name of a storage system
sessionId	unique identifier for each session for using the data handling system
sessionLength	maximum time in minutes for holding a session open
Size	size of container in bytes
SQL	stands for SQL control sequence that is registered as a digital entity
srbHost	specifies the SRB server that will be used to access the metadata catalog
srbPort	specifies the port number that will be used for a collection
targetCollName	the logical name of the collection/sub-collection under which the specified collection of digital entity is added
targetDataName	the new logical name for a digital entity
ticket	a unique identifier for tickets used for asynchronous authentication control
URL	specifies a URL that will be registered as a digital entity
userAddr	U.S. mail address of a user
userDomain	group to which the user will belong
userEmail	E-mail address of a user
username	unique name of a user
userPhone	telephone number of a user