DATA GRID MANAGEMENT SYSTEMS

Reagan W. Moore, Arun Jagatheesan, Arcot Rajasekar, Michael Wan, Wayne Schroeder
San Diego Supercomputer Center
9500 Gilman Drive, MC 0505
La Jolla, CA 92093
Tel: +1-858-534-5000, Fax: +1- 858-534-5152
e-mail: {moore,arun,sekar,mwan,schroede}@sdsc.edu

Abstract:
The "Grid" is an emerging infrastructure for coordinating access across autonomous organizations to distributed, heterogeneous computation and data resources. Data grids are being built around the world as the next generation data handling systems for sharing, publishing, and preserving data residing on storage systems located in multiple administrative domains. A data grid provides logical namespaces for users, digital entities and storage resources to create persistent identifiers for controlling access, enabling discovery, and managing wide area latencies. This paper introduces data grids and describes data grid use cases. The relevance of data grids to digital libraries and persistent archives is demonstrated, and research issues in data grids and grid dataflow management systems are discussed.

1 Introduction

A major challenge in the design of a generic data management system is the set of multiple requirements imposed by user communities. The amount of data is growing exponentially, both in the number of digital entities and in the size of files. The sources for data are distributed across multiple sites, with data generated in multiple administration domains, and on sites only accessible over wide-area networks. The need for discovery is becoming more important, with data assembled into collections that can be browsed. The need for preservation is becoming more important, both to meet legal data retention requirements, and to preserve the intellectual capital of organizations. In practice, six types of data handling systems are found:

1. Data ingestion systems
2. Data collection creation environments
3. Data sharing environments based on *data grids*
4. Digital libraries for publication of data
5. Persistent archives for data preservation
6. Data processing pipelines

The goal of a generic data management system is to build software infrastructure that can meet the requirements of each of these communities. We will demonstrate that data grids provide the generic data management abstractions needed to manage distributed data, and that all of the systems can be built upon common software.

Data management requires four basic naming conventions (or information categories) for managing data on distributed resources:

1. Resource naming for access to storage systems across administrative domains. This is used to implement data storage virtualization.
2. Distinguished user names for identifying persons across administrative domains. This is used to implement single-sign on security environments.
3. Distinguished file names for identifying files across administrative domains. This is used to implement data virtualization.
4. Context attributes for managing state information generated by remote processes. This is used to implement digital libraries and federate data grids.

A data grid [1,2] provides virtualization mechanisms for resources, users, files, and metadata. Each virtualization mechanism implements a location and infrastructure independent name space that provides persistent identifiers. The persistent identifiers for data are organized as a collection hierarchy and called a "*logical name space*". In practice, the logical name spaces are implemented in a separate metadata catalog for each data grid. Within a data grid, access to data, management of data, and manipulation of data is done via commands applied to the logical name space.

To access data located within another data grid (another logical name space), federation mechanisms are required. To manage operations on the massive collections that are assembled, data flow environments are needed. We illustrate the issues related to data grid creation, data management, data processing, and data grid federation by examining how these capabilities are used by each of the six types of data management systems. We then present the underlying abstraction mechanisms provided by data grids, and close with a discussion of current research and development activities in data flow and federation infrastructure.

2 Data Management Systems

Data management systems provide unifying mechanisms for naming, organizing, accessing, and manipulating context (administrative, descriptive, and preservation metadata) about content (digital entities such as files, URLs, SQL command strings, directories). Each of the types of data management system approaches focuses on a different aspect, and provides specific mechanisms for data and metadata manipulation.

2.1 Data ingestion systems

The Real-time Observatories, Applications, and Data management Network (ROADNet) [3] project manages ingestion of data in real-time from sensors. The data is assembled both synchronously and asynchronously from multiple networks into an object ring buffer (ORB), where it is then registered into a data grid. Multiple object ring buffers are federated into a Virtual Object Ring Buffer (VORB) to support discovery and attribute-based queries. Typical operations are the retrieval of the last ten observations, the tracking of observations about a particular seismic event, and the migration of data from the ORB into a remote storage system.

A second type of data ingestion system is a grid portal, which manages interactions with jobs executing in a distributed environment. The portal provides access to collections for input data, and stores output results back into a collection. A grid portal uses the Grid Security Infrastructure to manage inter-realm authentication between compute and storage sites.

2.2 Data collection creation environments

Scientific disciplines are assembling data collections that represent the significant digital holdings within their domain. Each community is organizing the material into a coherent collection that supports uniform discovery semantics, uniform data models and data formats, and an assured quality. The National Virtual Observatory (NVO) [4] is hosting multiple sky survey image collections. Each collection is registered into a logical name space to provide a uniform naming convention, and standard metadata attributes are used to describe the sky coverage of each image, the filter that was used during observation, the date the image was taken, etc. Collection formation is facilitated by the ability to register the descriptive metadata attributes onto a logical name space. The logical name space serves as the key for correlating the context with each image.

Other examples include: GAMESS [5], computational chemistry data collections of simulation output; and the CEED: Caveat Emptor Ecological Data Repository. Both projects are assembling collections of data for their respective communities.

2.3 Data sharing environments based on data grids

Scientific disciplines promote the sharing of data. While collections are used to organize the content, data grids are used to manage content that is distributed across multiple sites. The data grid technology provides the logical name space for registering files, the inter-realm authentication mechanisms, the latency management mechanisms, and support for high-speed parallel data transfers. An example is the Joint Center for Structural Genomics data grid which generates crystallographic data at the Stanford Linear Accelerator and pushes the data to SDSC for storage in an archive and analysis of protein structures. The Particle Physics Data Grid [6] federates collections housed at Stanford and Lyon, France for the BaBar high energy physics experiment [7]. The Biomedical Informatics Research Network (BIRN) [8] is using a data grid to share data from multiple Magnetic Resonance Imaging laboratories. Each project implements access controls that are applied on the distributed data by the data grid, independently of the underlying storage resource.

2.4 Digital libraries for publication of data

An emerging initiative within digital libraries is support for standard digital reference sets that can be used by an entire community. The standard digital reference sets are created from observational data collections, or from simulations, and are housed within the digital library. Curation methods are applied to assure data quality. Discovery mechanisms are supported for attribute-based query. An example is the HyperAtlas catalog that is being created for the 2MASS and DPOSS astronomy sky surveys [4,19,20]. The catalog projects each image to a standard reference frame, organizes the

projections into an atlas of the sky, and supports discovery through existing sky catalogs of stars and galaxies.

The organizations participating in a collaboration may share their digital entities using a collective logical view. Multiple logical views could be created for the same set of distributed digital entities. These logical views may be based on different taxonomies or business rules that help in the categorization of the data. The organizations, apart from sharing the physical data, could also share the logical views as publication mechanisms.

The library community applies six data management processes to digital entities:

1. Collection building to organize digital entities for access
2. Content management to store each digital image
3. Context management to define descriptive metadata
4. Curation processes to validate the quality of the collection
5. Closure analyses to assert completeness of the collection and the ability to manipulate digital entities within the collection
6. Consistency processes to assure that the context is updated correctly when operations are performed on the content.

2.5 Persistent archives for data preservation

The preservation community manages archival collections for time periods that are much longer that the lifetimes of the underlying infrastructure [9,10,11,25,26,27]. The principal concern is the preservation of the authenticity of the data, expressed as an archival context associated with each digital entity, and the management of technology evolution. As new more cost effective storage repositories become available, and as new encoding formats appear for data and metadata, the archival collection is migrated to the new standard. This requires the ability to make replicas of data on new platforms, provide an access abstraction to support new access mechanisms that appear over time, and migrate digital entities to new encoding formats or emulate old presentation applications. Example projects include a persistent archive for the National Science Digital Library, and a persistent archive prototype for the National Archives and Records Administration [12.13]. Both projects manage data that is stored at multiple sites, replicate data onto different types of storage repositories, and support both archival and digital library interfaces.

The preservation community applies their own standard processes to data:

1. Appraisal – the decision for whether a digital entity is worth preserving
2. Accession – the controlled process under which digital entities are brought into the preservation environment
3. Arrangement – the association of digital entities with a record group or record series
4. Description – the creation of an archival context specifying provenance information
5. Preservation – the creation of an archival form for each digital entity and storage
6. Access – the support for discovery services

2.6 Data Processing Pipelines

The organization of collections, registration of data into a data grid, curation of data for ingestion into a digital library, and preservation of data through application of archival processes, all need the ability to apply data processing pipelines. The application of processes is a fundamental operation needed to automate data management tasks. Scientific disciplines also apply data processing pipelines to convert sensor data to a standard representation, apply calibrations, and create derived data products. Examples are the Alliance for Cell Signaling digital library, which applies standard data analysis techniques to interpret each cell array, and the NASA Earth Observing Satellite system that generates derived data products from satellite observations.

Data processing pipelines are also used to support knowledge generation. The steps are:

1. Apply semantic labels to features detected within a digital entity
2. Organize detected features for related digital entities within a collection
3. Identify common relationships (structural, temporal, logical, functional) between detected features
4. Correlate identified relationships to physical laws

3 Generic requirements

Multiple papers that summarize the requirements of end-to-end applications have been generated in the Global Grid Forum [14]. They range from descriptions of the remote operations that are needed to manage large collections, to the abstraction mechanisms that are needed for preservation. The capabilities can be separated into four main categories: Context management, data management, access mechanisms, and federation mechanisms. The capabilities are provided by data grids:

Context management mechanisms

- Global persistent identifiers for naming files.
- Organization of context as collection hierarchy
- Support for administrative metadata to describe the location and ownership of files
- Support for descriptive metadata to support discovery through query mechanisms
- Support for browsing and queries on metadata
- Information repository abstraction for managing collections in databases

Data management mechanisms

- Storage repository abstraction for interacting with multiple types of storage systems
- Support for the registration of files into the logical name space
- Inter-realm authentication system for secure access to remote storage systems
- Support for replication of files between sites
- Support for caching onto a local storage system and for accessing files in an archive
- Support for aggregating files into containers

Access mechanisms

- Standard access mechanisms: Web browsers, Unix shell commands, Windows browsers, Python scripts, Java, C library calls, Linux I/O redirection, WSDL, etc.
- Access controls and audit trails to control and track data usage

- Support for the execution of remote operations for data sub-setting, metadata extraction, indexing, third-party data movement, etc.
- Support for bulk data transfer of files, bulk metadata transfer, and parallel I/O

Federation mechanisms

- Cross-registration of users between data grids
- Cross-registration of storage resources data grids
- Cross-registration of files between data grids
- Cross-registration of context between data grids

Data Grids provide transparency and abstraction mechanisms that enable applications to access and manage data as if they were local to their home system. Data grids are implemented as federated client-server middleware that use collections to organize distributed data.

4 Data Grid Implementation

An example of a data grid is the Storage Resource Broker (SRB) from the San Diego Supercomputer Center [15,16,17]. The SRB manages context (administrative, descriptive, and preservation metadata) about content (digital entities such as files, URLs, SQL command strings, directories). The content may be distributed across multiple types of storage systems across independent administration domains. By separating the context management from the content management, the SRB easily provides a means for managing, querying, accessing, and preserving data in a distributed data grid framework. Logical name spaces describe storage systems, digital file objects, users, and collections. Context is mapped to the logical name spaces to manage replicas of data, authenticate users, control access to documents and collections, and audit accesses. The SRB manages the context in a Meta data Catalog (MCAT) [18], organized as a collection hierarchy. The SRB provides facilities to associate user-defined metadata, both free-form attribute-based metadata and schema-based metadata at the collection and object level and query them for access and semantic discovery. The SRB supports queries on descriptive attributes [21]. The SRB provides specific features needed to implement digital libraries, persistent archive systems [10,11,12,13] and data management systems [22,23,24].

The Storage Resource Broker (SRB) in earlier versions used a centralized MCAT [18] for storing system-level and application-level metadata. Though essential for consistent operations, the centralized MCAT poses a problem. It can be considered a single-point of failure as well as a potential bottleneck for performance. Moreover, when users are widely distributed, users remote from the MCAT may see latency unacceptable for interactive data access.

In order to mitigate these problems, the SRB architecture has been extended to a federated environment, called zoneSRB. The ZoneSRB architecture provides a means for multiple context catalogs to interact with each other on a peer-to-peer basis and synchronize their data and metadata. Each zoneSRB system can be autonomous, geographically distant, and administer a set of users, resources and data that may or may

not be shared by another zoneSRB. Each zoneSRB has its own MCAT for providing the same level of features and facilities as done by the older SRB system.

The main advantage of the zoneSRB system is that now, there is no single point of failure, as multiple MCATs can be federated into a multi-zoneSRB system. Users can be distributed across the zones to improve quality of performance and minimize access latencies to geographically distant metadata catalogs. The multiple zoneSRBs can share metadata and data based on policies established by the collaborating administrators. The level of collaboration can be varied to specify how much of the information is shared, partitioned or overlapped, and whether the interactions are controlled by the users or the zone administrators.

More information about the SRB can be found in [15,16,17,18]. In a nutshell, the SRB provides all of the capabilities listed as generic requirements. The SRB provides interoperability mechanisms that map users, datasets, collections, resources and methods to global namespaces. It also provides abstractions for data management functionality such as file creation, access, authorization, user authentication, replication and versioning, and provides a means to associate metadata and annotation with data objects and collections of data objects. Descriptive metadata is used for searching at the semantic level and discovery of relevant data objects using the attribute-based discovery paradigm. Figure 1 provides details about the modules that form the core of the SRB services.

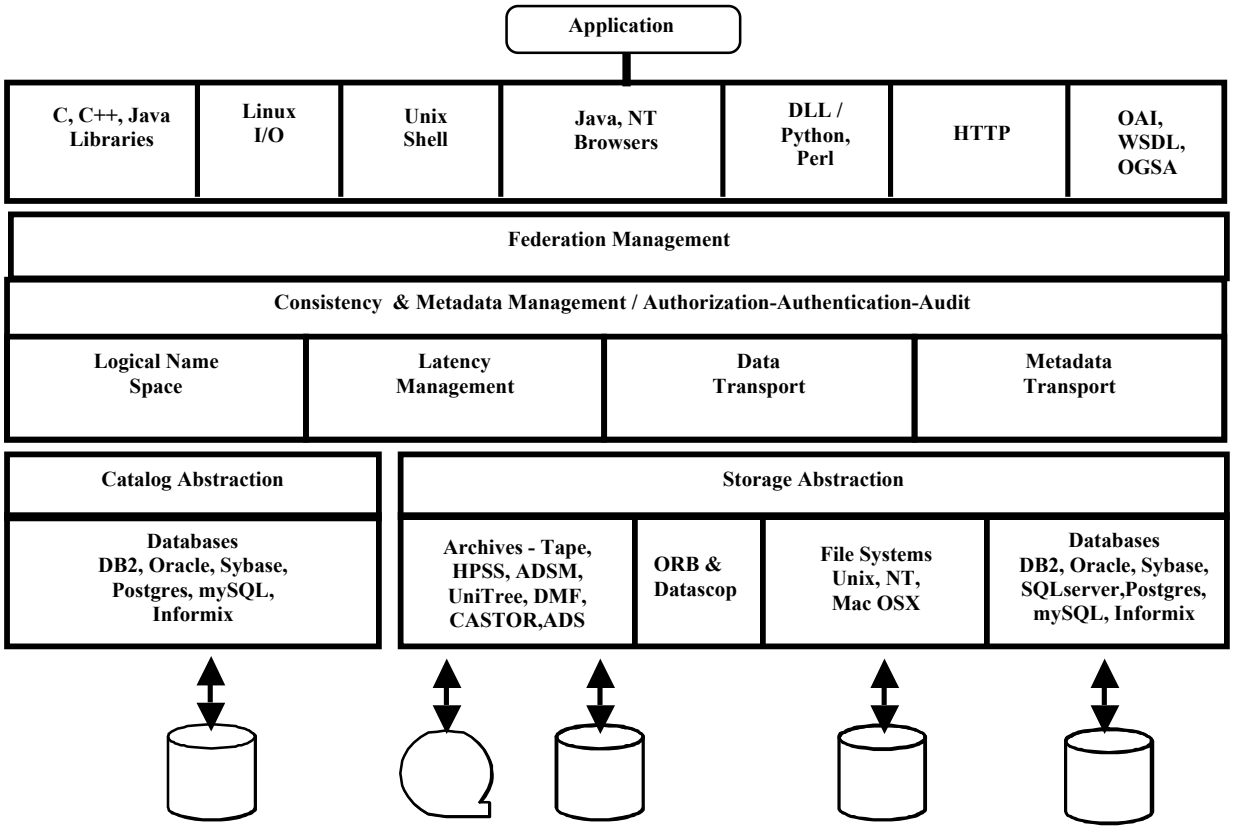

Figure 1. Storage Resource Broker

The SRB server is built using three layers. The top-layer is a set of server access APIs written in C and Java that are used to provide multiple client interfaces. The middle layer provides the intelligence for collection-based management, federation of data grids, data movement, authentication, and authorization. The bottom-layer is a set of storage drivers and MCAT database drivers that are used to connect to diverse resources. These drivers have a well-defined and published API such that a new storage resource or data server system can be integrated easily into the SRB system. For example, custom-based interfaces to special drivers such as the Atlas Data Store System and CERN's CASTOR storage systems were written in just a few days. The SRB drivers for storage include, HPSS, ADSM, Unix/Linux /Mac OSX and NT file systems, and for databases includes DB2, Oracle, Postgres, Informix, and Sybase.

The middle layer of the SRB system supports the federation consistency and control mechanisms needed to integrate multiple data grids and encapsulates much of the intelligence needed to manage a data grid. This layer interacts with the MCAT to access the context needed to control each type of data service.

The data grid technology based on the SRB that is in production use at SDSC at this date manages over 90 Terabytes of data comprising over 16 million files.

5 Data Grid Federation

The specification of appropriate federation mechanisms is still a research project. The federation of multiple data grids basically imposes constraints on the cross-registration of users, resources, files, and context. The constraints may be invoked by either a user or automatically implemented by the data management system. The constraints may be set for either no cross-registration, partial cross-registration, or complete cross-registration. It is possible to identify over 1500 possible federation approaches by varying the type of cross-registration constraints that are imposed. In practice, ten of the approaches are either in active use or proposed for use within scientific projects. Each data grid is called a zone, with its own metadata catalog managing the logical name spaces for its users, resources, files, and context. The approaches include:

5.1 Occasional Interchange

This is the simplest model in which two or more zones operate autonomously with very little exchange of data or metadata. The two zones exchange only user-ids for those users who need access across zones. Most of the users stay in their own zone accessing resources and data that are managed by their zone MCAT. Inter-zone users will occasionally cross zones, browsing collections, querying metadata and accessing files for which they have read permission. These users can store data in remote zones if needed but these objects are only accessible to users in the other zones. This model provides the greatest degree of autonomy and control. The cross-zone user registration is done not for every user from a zone but only for selected users. The local SRB administrator controls who is given access to the local SRB system and can restrict these users from creating files in the local SRB resources. (NPACI driven federation model [28])

5.2 **Replicated Catalog**

In this model, even though there are multiple MCATs managing independent zones, the overall system behaves as though it were a single zone with replicated MCATs. Metadata about the tokens being used, users, resources, collections, containers and data objects are all synchronized between all MCATs. Hence, the view from every zone is the same. An object created in a zone is registered as an object in all other sister zones and any associated metadata is also replicated. This model provides a completely replicated system that has a high degree of fault-tolerance for MCAT failures. The user can still access data even if their local MCAT becomes non-functional. The degree of synchronization, though very high in principle, in practice is limited. The MCATs may be out of synchronization on newly created data and metadata. The periodicity of synchronization is decided by the cooperating administrators and can be as long as days if the systems change slowly. An important point to note is that because of these delayed synchronizations, one might have occasional logical name clashes. For example, a data object with the same name and in the same collection might be created in two zones almost at the same time. Because of delayed synchronization both will be allowed in their respective zones. But when the synchronization is attempted, the system will see a clash when registering across zones. The resolution of this has to be done by mutual policies set by the cooperating administrators. In order to avoid such clashes, policies can be instituted with clear lines of partitioning about where one can create a new file in a collection. (NARA federation model [12])

5.3 **Resource Interaction**

In this model resources are shared by more than one zone and hence they can be used for replicating data. This model is useful if the zones are electronically distant, but want to make it easier for users in the sister zone to access data that might be of mutual interest. A user in a zone replicates data into the shared resources (either using synchronous replication or asynchronous replication as done in a single zone). Then the metadata of these replicated objects is synchronized across the zones. User names need not be completely synchronized. (BIRN federation model [8])

5.4 **Replicated Data Zones**

In this model two or more zones work independently but maintain the same data across zones, i.e., they replicate data and related metadata across zones. In this case, the zones are truly autonomous and do not allow users to cross zones. In fact, user lists and resources are not shared across zones. But data stored in one zone is copied into another zone along with related metadata, by a user who has accounts in the sister zones. This method is very useful when two zones are operating across a wide-area network has to share data and the network delay in accessing data across the zones has to be reduced. (BaBar federation model [7])

5.5 **Master-Slave Zones**

This is a variation of the 'Replicated Data Zones' model in which new data is created at a Master site and the slave sites synchronize with the master site. The user list and resource list are distinct across zones. The data created at the master are copied over to the slave zone. The slave zone can create additional derived objects and metadata but these may

not be shared back to the Master Zone. (PDB federation model)

5.6 Snow-Flake Zones

This is a variation of the 'Master-Slave Zones' model, One can view this as a hierarchical model, where a Master Zone creates the data that is copied to the slave zones, whose data in turn gets copied into other slave zones lower in the hierarchy. Each level of the hierarchy can create new derived products of data and metadata, can have their own client base, and can choose to propagate only a subset of their holdings to their slave zones. (CMS federation model [23]).

5.7 User and Data Replica Zones

This is another variation of the 'Replicated Data Zones' where not just the data get replicated but also user names are exchanged. This model allows users to access data in any zone. This model can be used for wide-area enterprises where users travel across zones and would like to access data from their current locations for improved performance. (Web cache federation model)

5.8 Nomadic Zones - SRB in a Box

In this model, a user might have a small zone on a laptop or other desktop systems that are not always connected to other zones. The user during his times of non-connectedness can create new data and metadata. The user on connecting to the parent zone will then synchronize and exchange new data and metadata across the user-zone and the parent zone. This model is useful for users who have their own zones on laptops. It is also useful for zones that are created for ships and nomadic scientists in the field who periodically synchronize with a parent zone. (SIOExplorer federation model)

5.9 Free-floating Zones – myZone

This is a variation of the 'Nomadic Zone' model having multiple stand-alone zones but no parent zone. These zones can be considered peers and possibly have very few users and resources. These zones can be seen as isolated systems running by themselves (like a PC) without any interaction with other zones, but with a slight difference. These zones occasionally "talk" to each other and exchange data and collections. This is similar to what happens when we exchange files using zip drives or CDs or as occasional network neighbors. This system has a good level of autonomy and isolation with controlled data sharing. (Peer-to-peer or Napster federation model)

5.10 Archival Zone, BackUp Zone

In this model, there can be multiple zones with an additional zone called the archive. The main purpose of this is to create an archive of the holdings of the first set of zones. Each zone in the first set can designate the collections that need to be archived. This provides for backup copies for zones which by themselves might be fully running on spinning disk. (SDSC backup federation model, NASA backup federation model [29])

6 Grid Dataflow

A second research area is support for data flow environments, in which state information is kept about the processing steps that have been applied to each digital entity in a work set.

6.1 Need for peer-to-peer Data Grid Dataflows

A dataflow executes multiple tasks. Each task might require: different resources; access to different data collections for input; storage of output products onto physically distributed resources within a data grid; and disparate services that might be in the form of web/grid services or simply executables of an application. The dataflow is described in a data grid language. The dataflow is executed through a dataflow engine. Each dataflow engine needs to be able to communicate with other dataflow engines in a peer-to-peer federation for coordination. This allows dynamic distributed execution, without having to specify a pre-planned schedule.

Placement scheduling is still required to find the right location for execution of each task. In the data grid, the tasks in the dataflow could be executed in any of distributed resources within the participating administrative domains. In general the following factors must be considered for dataflow scheduling:

- *Appropriateness of a given resource for a particular task:* Is there enough disk space to hold result sets, and is the compute resource powerful enough to execute the task within a desired time? Are the tasks sufficiently small that they could be processed by less powerful systems?
- *Management of data movement:* How can the amount of data moved for both input and output files and for the executable be minimized?
- *Co-location of dependent tasks:* How can tasks be co-located on the same administrative domain or resource to minimize coordination messages that have to be sent across the network?

6.2 Grid Dataflow System Requirements

Collections of data sets can be manipulated in a Data Grid dataflow. Instead of creating a separate dataflow for each file, state information can be maintained about the aggregated set of files for which processes have been applied. Related issues are:

- *Management of processing state*: (e.g.) What information needs to be maintained about each process?
- *Control procedures*: (e.g.) What types of control mechanisms are needed to support loops over collections?
- *Dynamic status queries*: (e.g.) Can a process detect the state of completion of other processes through knowledge of the placement schedule?

6.3 Data Grid Language

The SDSC Matrix project [33], funded by the NSF Grid Physics Network (GriPhyN) [30], NIH Biomedical Informatics Research Network (BIRN) [8] and NSF Southern California Earthquake Center (SCEC) [31], has developed a data grid language to describe grid dataflow. Just like SQL (Structured Query Language) is used to interact with the databases, the Data Grid Language (DGL) is used to interact with the data grids

and dataflow environments. DGL is XML-based and uses a standard schema that describes:

- Control-based dataflow structures. These include sequential, parallel, and aggregated process execution.
- Context-based dataflow structures. These include barriers (synchronization points or milestones), "For loops" (iteration over task sets) and "For Each loops" (iteration over collection of files).
- Event Condition Alternate Action (ECA) rules. Any event in the workflow engine like completion or start of a task could be used to trigger a condition to be evaluated dynamically and execute any of the alternate dataflow actions. The conditions could be described using XQuery or any other language that would be understood by the dataflow engine. This allows other useful or simple workflow query languages to be used along with DGL.
- Variables. Both global variables and local variables can be managed for the dataflow. The variables are related to the dataflow, rather than an individual file that is manipulated by the dataflow. Hierarchical scoping is used to restrict the use of the dataflow variables to aggregates of processes.
- Discovery. Queries from external grid processes are supported for determining the completion status of a process and the state of variables.

A simple example of the use of dataflow systems is the management of the ingestion of a collection into a data grid. The SCEC project implemented the collection ingestion as a dataflow using the data grid language and executed the dataflow using a SDSC Matrix Grid workflow engine.

6.4 SDSC Matrix Architecture

The architecture of the SDSC Matrix dataflow engine is shown in Figure 2. The components are layered on top of agents that can execute either SRB or other processes (SDSC Data Management Cyberinfrastructure, java classes, WSDL services and other executables). The matrix dataflow engine tracks the dataflow execution. The dataflow execution state can be queried by other applications or other dataflows that are executed by the matrix engine. Persistence of the dataflow execution state is held in memory and exported to a relational database.

Clients send DGL dataflow requests as SOAP [32] messages to the Java XML (JAXM) messaging interface (Fig 2). The Matrix web service receives these SOAP messages and forwards the DGL requests to the Data Grid Request Processor. The Request Processor parses the DGL requests, which could be either a *data grid transaction* (new dataflow) or a *status query* on another dataflow. A data grid transaction request is a long running dataflow involving the execution of multiple data management and compute intensive processes. The *Transaction Handler* registers a new transaction and starts the book keeping and execution of the processes. The *Status Query Handler* is used to query the state of execution of the transaction and the variables associated with a dataflow. In Figure 2, the Matrix Engine components shown in white boxes have been implemented for a stand-alone matrix workflow engine (version 3.1). Those in solid (black) boxes are a work in progress to provide peer-to-peer grid workflow and involve protocols to distribute the workflow. The P2P broker will be based on Sangam protocols [34] to

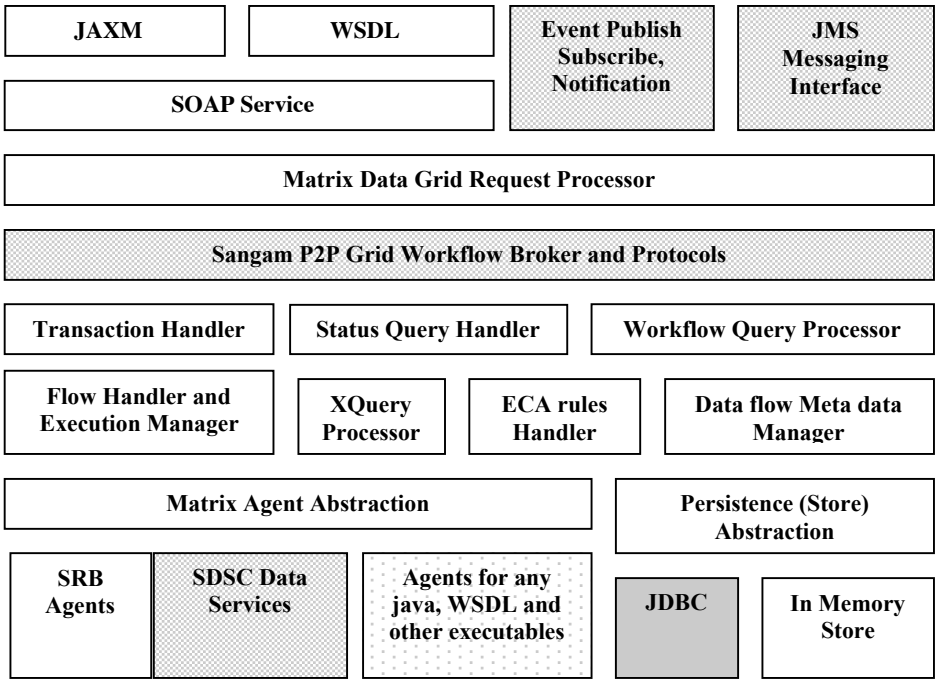

Figure 2. SDSC Matrix Grid Workflow Engine Architecture for data flows

facilitate Peer-to-peer brokering between workflow engines. The protocols will be loosely coupled with resource scheduling algorithms.

7 Conclusion

Data grids provide a common infrastructure base upon which multiple types of data management environments may be implemented. Data grids provide the mechanisms needed to manage distributed data, the tools that simplify automation of data management processes, and the logical name spaces needed to assemble collections. The Storage Resource Broker data grid is an example of a system that has been successfully applied to a wide variety of scientific disciplines for management of massive collections. Current research issues include identification of the appropriate approaches for federating data grids, and the development of capable data flow processing systems for the management of data manipulation.

8 Acknowledgement

The results presented here were supported by the NSF NPACI ACI-9619020 (NARA supplement), the NSF NSDL/UCAR Subaward S02-36645, the NSF Digital Library Initiative Phase II Interlib project, 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 authors would also like acknowledge other members of the SDSC SRB team who have contributed to this work including: George Kremenek, Bing Zhu, Sheau-Yen Chen, Charles Cowart, Roman Olschanowsky, Vicky Rowley and Lucas Gilbert. The members

of the SDSC Matrix Project include Reena Mathew, Jon Weinberg, Allen Ding and Erik Vandekieft.

9 References

[1] Foster, I., and Kesselman, C., (1999) "The Grid: Blueprint for a New Computing Infrastructure," Morgan Kaufmann.

[2] Rajasekar, A., M. Wan, R. Moore, T. Guptill, "Data Grids, Collections and Grid Bricks," Twentieth IEEE/Eleventh NASA Goddard Conference on Mass Storage Systems & Technologies, April 7-10, 2003, San Diego, USA.

[3] Real-time Observatories, Applications, and Data management Network (RoadNet), http://roadnet.ucsd.edu/

[4] NVO, (2001) "National Virtual Observatory", (http://www.srl.caltech.edu/nvo/).

[5] GAMESS "General Atomic Molecular Electronic Structure Systems - Web Portal. (https://gridport.npaci.edu/GAMESS/).

[6] PPDG, (1999) "The Particle Physics Data Grid", (http://www.ppdg.net/, http://www.cacr.caltech.edu/ppdg/).

[7] BABAR Collaboration (B. Aubert et al.)., "The First Year of the Babar Experiment At PEP-II. 30th International Conference on High-Energy Physics (ICHEP 2000), Japan.

[8] BIRN, "The Biomedical Informatics Research Network", http://www.nbirn.net

[9] Rajasekar, A., R. Marciano, R. Moore, (1999), "Collection Based Persistent Archives," Proceedings of the 16th IEEE Symposium on Mass Storage Systems, 1999.

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

[11] Moore, R., A. Rajasekar, "Common Consistency Requirements for Data Grids, Digital Libraries, and Persistent Archives", Grid Protocol Architecture Research Group draft, Global Grid Forum, April 2003.

[12] US National Archives and Records Administration, http://www.archives.gov/, also see http://www.sdsc.edu/NARA/

[13] Moore, R., C. Baru, A. Gupta, B. Ludaescher, R. Marciano, A. Rajasekar, (1999), "Collection-Based long-Term Preservation," GA-A23183, report to National Archives and Records Administration, June, 1999.

[14] GGF, "The Global Grid Forum" (http://www.ggf.org/)

[15] SRB, "Storage Resource Broker Website", SDSC (http://www.npaci.edu/dice/srb).

[16] Rajasekar, A., Wan, M., Moore, R.W., Schroeder, W., Kremenek, G., Jagatheesan, A., Cowart, C., Zhu, B., Chen, S.Y. and Olschanowsky, R., "Storage Resource Broker – Managing Distributed Data in a Grid," *Computer Society of India Journal, special issue on SAN*, 2003

[17] Rajasekar, A., Wan, M., Moore, R.W., Jagatheesan, A. and Kremenek, G., "Real Experiences with Data Grids – Case-studies in using the SRB," Proceedings of *6th International Conference/Exhibition on High Performance Computing Conference in Asia Pacific Region (HPC-Asia)*, December 2002, Bangalore, India

[18] MCAT - "The Metadata Catalog", http://www.npaci.edu/DICE/SRB/mcat.html

[19] 2-Micron All Sky Survey (2MASS), http://www.ipac.caltech.edu/2mass/

[20] Digital Palomar Observatory Sky Survey, http://www.astro.caltech.edu/~george/dposs/

[21] Moore R. and A. Rajasekar, "Data and Metadata Collections for Scientific Applications," High Performance Computing and Networking, Amsterdam, NL, 2001.
[22] Wan, M., A. Rajasekar, R. Moore, "A Simple Mass Storage System for the SRB Data Grid," Twentieth IEEE/Eleventh NASA Goddard Conference on Mass Storage Systems & Technologies, April 7-10, 2003, San Diego, USA.
[23] Hoschek, W., Jaen-Martinez, J., Samar, A., Stockinger, H., and Stockinger, K. (2000) "Data Management in an International Data Grid Project," *IEEE/ACM International Workshop on Grid Computing Grid'2000*, Bangalore, India 17-20 December 2000. (http://www.eu-datagrid.org/grid/papers/data_mgt_grid2000.pdf).
[24] 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.
[25] Thibodeau, K., "Building the Archives of the Future: Advances in Preserving Electronic Records at the National Archives and Records Administration", U.S. National Archives and Records Administration,
http://www.dlib.org/dlib/february01/thibodeau/02thibodeau.html
[26] Underwood, W. E., "As-Is IDEF0 Activity Model of the Archival Processing of Presidential Textual Records," TR CSITD 98-1, Information Technology and Telecommunications Laboratory, Georgia Tech Research Institute, December 1, 1988.
[27] Underwood, W. E., "The InterPARES Preservation Model: A Framework for the Long-Term Preservation of Authentic Electronic Records". Choices and Strategies for Preservation of the Collective Memory, Toblach/Dobbiaco Italy 25-29 June 2002, Archivi per la Storia.
[28] NPACI Data Intensive Computing Environment thrust, http://www.npaci.edu/DICE/
[29] NASA Information Power Grid (IPG), (http://www.ipg.nasa.gov/)
[30] GriPhyN, "The Grid Physics Network", (http://www.griphyn.org/).
[31] SCEC Web Site, Southern California Earthquake Center, (http://www.scec.org/)
[32] Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H.F., Thatte, S., and Winer, D., *"Simple Object Access Protocol (SOAP)"* W3C Note.
http://www.w3.org/TR/SOAP/
[33] SDSC Matrix Project Web site http://www.npaci.edu/DICE/SRB/matrix/
[34] Jagatheesan, A., "Architecture and Protocols for Sangam Communities and Sangam E-Services Broker," Technical Report, (Master's Thesis) CISE Department, University of Florida, 2001 http://etd.fcla.edu/UF/anp1601/ArunFinal.pdf
[35] Helal, A., Su, S.Y.W., Meng, Jei, Krithivasan, R. and Jagatheesan, R., "The Internet Enterprise," Proceedings of Second IEEE/IPSJ Symposium on Applications and the Internet (SAINT 02), February 2002, Japan.
http://www.harris.cise.ufl.edu/projects/publications/Internet_Enterprise.pdf