Policy-based Distributed Data Management Systems

Arcot Rajasekar
Reagan Moore
University of North Carolina at Chapel Hill

Mike Wan
Wayne Schroeder
University of California, San Diego

Abstract
Scientific research collaborations generate massive amounts of data that are assembled into collections, published in digital libraries, processed in data analysis pipelines, and preserved in reference collections. Policy-based data management systems minimize the amount of labor needed to manage the massive collections by automating the enforcement of management policies and the validation of assessment criteria. The goal is data management infrastructure that can be used to support all phases of the data life cycle, while minimizing the amount of labor needed to maintain the collection.

1. Introduction:

With the ongoing explosion of data creation in digital form, digital repositories are being deployed in large-scale virtual environments [1]. Such repositories allow data sharing across research communities as well as provide information for public users. An important aspect of such open repositories is that they are geographically dispersed operating under autonomous administrative and disciplinary boundaries. Such repositories also interact with distributed computational services (such as super computer centers [2] and cloud computing providers [3]) and disperse information through collaborating social communities and portals [4]. The requirement to provide a unified *open repository framework* -integrating and spanning multiple autonomous repositories - leads to challenges that encompass strategic problems in data management across the entire lifecycle of information - from supporting the creation and management of digital content, to enabling use, re-use, and interconnection of information, to ultimately ensuring long- term preservation and archiving.

Multiple open repositories are being assembled by scientific and humanities groups. They are bringing together data from distributed researchers in order to provide a single uniform accessible portal that can be used by researchers and public alike. Some of the national and international efforts with these goals are as follows:

- Astronomical data: The National Virtual Observatory [5] is assembling standard mechanisms for sharing catalog information and sky survey images. They have established interoperability mechanisms for querying catalogs and for accessing data within a storage system.
- Oceanographic data: The Ocean Observations Initiative [6] is exploring the integration of cloud computing systems and cloud storage caches with institutional repositories. One goal is to manage extraction of previous observations from an archive, caching of the data on a cloud resource, and on –demand analysis of the data.

- Plant data: The iPlant Collaborative [7] is federating existing data collections to create a community-driven research environment. One goal is to enable multi-disciplinary research and manage data re-use across disciplines.
- Science of Learning Centers: The NSF Science of Learning Centers [8] support six research areas in cognitive science. They have the challenge of both sharing data within a research area, and sharing data between research areas.
- Biomedical data: The Biomedical Informatics Research Network [9] is building a public data repository for sharing brain images.
- Hydrological data: The Consortium of Universities for the Advancement of Hydrologic Science (CUAHSI) organizes point observation data into a shareable resource and provides tools for displaying and analyzing water data [10].
- Earth Systems Data: DataOne[11] is an NSF datanet initiative that will provide universal access to data about life on earth and the environment.
- Scientific Computation: The NSF TeraGrid manages simulation output, providing both high-performance access for post-processing, as well as a long-term archive.
- HASTAC: The Humanities, Arts, Science and Technology Advanced Collaboratory promotes new platforms for social interaction.
- Odum Institute for Research in Social Science maintains an archive of computer-readable social science data, and supports discovery and controlled access to the collections.
- ARCS: The Australian Research Collaboration Service provides long-term eResearch support through federation of shared collections across research institutions in Australia.
- NARA TPAP: The National Archives and Records Administration Transcontinental Persistent Archive Prototype is a research platform for investigating preservation principles.
- NCCS: The NASA Center for Computational Science provides simulation tools and organizes the output into shareable collections.
- UK eScience data grid builds shared collections and manages an archive for research results.

Additional large scale data sharing initiatives are under design and development. These include:

- LSST: The Large Synoptic Survey Telescope is developing a data grid to manage transport of 150 petabytes of images from a telescope in Chile to the US, analysis of the images, and archiving of the data.
- NCDC: The National Climatic Data Center is building a data grid to manage 150 petabytes of satellite images of the earth. The data grid will link computing resources with data archives.
- CERN LHC: The Large Hadron Collider experiment will generate 15 petabytes of data per year, and distribute the data around the world to data analysis platforms.

The common concerns of these projects are for immediate sharing and discovery of data among collaborating researchers and to provide reference collections for long-term preservation to enable future research. Because of the challenges in integrating data from diverse projects (working in the same discipline and hence having an imperative to share) a coherent open repository framework is necessary. We describe such a framework, developed within our group, which is being used by several research initiatives for sharing data and also being evaluated by other groups for eventual adaptation.

Our approach to information lifecycle management for building an open repository framework is based on policy-oriented data management. Our thesis is that different stages in an open repository lifetime can be realized by a sequence of policy applications. These discrete policy sets in turn abide by the requirements of individual repositories that form the collaboratory framework, encode trust relationships among these repositories and enable smooth interactions among repositories as well as collaborating social communities. The outcome of this thesis is the development of a software framework called the *integrated Rule-Oriented Data System (iRODS)* [12] being developed by the Data Intensive Cyber Environments (DICE) group [13] with collaborations from various groups and projects all around the world. The iRODS system is a data grid [14] that spans geographically remote sites and supports a policy-oriented life cycle management for digital artifacts.

In the next section, we briefly discuss the iRODS system, and in Section 3 we develop the concept of an iRODS Open Repository framework. In Section 4, we show an exemplar open repository that is being developed by the Temporal Dynamics of Learning Center, and conclude in section 5.

2. The iRODS System

The iRODS [15] data grid can be viewed (see Figure 1) as a network of completely connected nodes of resource servers, called iRES servers that provide access to data and computational resources. These servers perform the protocol interchange that is needed for interfacing with exotic devices and map their protocols onto a uniform API that is used by the client framework.

The iRES servers are the workhorse of the grid and perform data movement between the servers and between the client and the servers. Also they are responsible for managing multiple types of data transfer modes (parallel, sequential, bundled, etc), multiple transfer protocols (XML, binary, TCP/IP, UDP, etc.), and operations related to manipulation of complex collections. iRES servers are also responsible for providing data management functionality and interfacing to other servers described below.

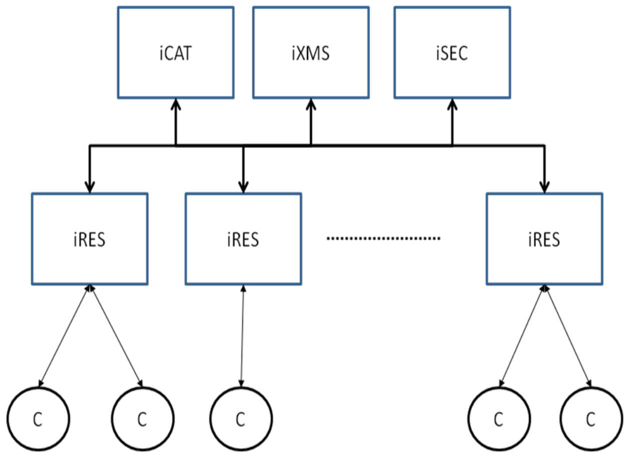

Figure 1. iRODS Components

In addition to the resource nodes there are three other special server nodes.

- The iCAT is a metadata catalog server that manages a relational database containing information (metadata) about the data sets, resources, users, servers,

rules and micro-services. Additional information required for authentication, authorization, auditing, accounting, etc. are also stored in the metadata catalog server. There is conceptually only one catalog server per data grid. The system allows one to have multiple catalog servers provided one is in the master mode and the rest are in the slave mode. The slave mode catalog servers are used for 'read only' operations and all catalog modifications are automatically directed to the master catalog server. The use of master-slave servers is mainly for load balancing and reducing access time when going across wide-area networks. All servers are aware of the location of the iCAT catalog server.
- The iXMS is a messaging server that provides a "mailbox" service with store and forward capability for messages between the server nodes and the micro-services that execute in these resources. This server node is used by the messaging micro-services to send, broadcast and receive messages. The server operates in both push and pull mode for message delivery. Even though there is no limitation to the number of messaging servers that can be operated in a grid, at least one should be operational and its address known to all other servers.
- The iSEC is a scheduler and execution server that can schedule operations on resource server nodes. This server, using information about pre-scheduled and queued rules and micro-services stored in the iCAT server, executes them when their execution time becomes valid. The server can also check for additional success conditions apart from time stamps. If all conditions are met, the server can execute the pre-scheduled action; otherwise the server puts the action back in the queue with retries being done based on options set by the user and/or the administrator.

The sequence of actions (micro-services or rules) that are performed during a rule execution can be viewed as a workflow. Since a rule can have more than one rule-definition, each of these rule-definitions can be viewed as an alternate "guarded" workflow with a priority imposed on the set of definitions. According to the iRODS implementation only one rule-definition "succeeds" when a rule is fired. Each alternative workflow is tried in order and if there is a failure of any of the alternative (either because the guard condition fails or because of a failure and recovery of the underlying workflow) the next rule in the priority list is invoked. If any one definition succeeds, the rule invocation is considered to be successful. If all rule-definitions fail then the rule invocation is a failure.

There are three kinds of micro-services:
1) System micro-services are the core services that are used for providing operational functions in the resource servers, iRES. These functions include low-level data management, data movement (e.g. data replication, copy, move), integrity and type checking (e.g. checksums such as MD5), collection-level operations, and micro-services for providing metadata management, messaging services and delayed execution and scheduling services. They also include services for authentication, authorization and auditing and for emailing users. Micro-services for user management, resource management and other administrative functions are also provided by system micro-services. The iRES high level operations (as invoked by the client) translate into a set of rule invocations that in turn invoke these micro-services.

2) Rule-Language Micro-services provide workflow control functions. The iRODS rule language does not provide powerful language constructs which one normally expects when building workflows, e.g. loops and conditional forks. Semantically (and in theory) these types of constructs can be coded using the simple rule-definition syntax by a rule developer. But it is more helpful and useful if syntactic constructs are pre-programmed to make it easier for rule programmers to code these complex workflows. The design decision in providing these functionalities was not to complicate the rule engine with complex rule interpretations, but to provide these functionalities as micro-services that are executed by the rule engine. The complexity of managing the workflow constructs such as loop variables and if-then-else conditionals is left to the design of the individual "rule-language' micro-services. We support constructs including while, for, forEachInList, ifThenElse, output strings and assignment. In addition, constructs for remote execution, parallel execution and delayed execution are also coded as micro-services that can be invoked. Each construct can execute a chain of micro services defined by input parameters.
3) Domain micro-services are domain-specific functions that are organized in modules of micro-services. A community can include as many modules as they need for their data system configuration to achieve a required functionality.

In iRODS, policies are encoded as rules of the form:

A :- $C \mid M_1, \ldots, M_n \mid R_1, \ldots, R_n$

where A is the name of the action (rule name),
C is the guard condition for the rule to fire
M_i is a micro-service or a rule, and
R_i is a recovery micro-service.

Micro-services in iRODS are well-defined C functions that take a set of arguments for input and output. A special argument called the 'white board' is also used for communicating between micro-services – one can view this as a global structure which gets passed intrinsically between the micro-services. Recovery micro-services are used when the sequence of micro-services fails at some point. The recovery micro-services are executed in order to roll back any changes or operations that were performed within the rule. The recovery micro-services provide a "transactional" capability for a rule such that either the whole rule is executed or the state is rolled back to a point before the execution of the rule. The semantics of the rules is that only one rule is "fully" executed on any invocation. All other rules of the same name that were tried whose conditions fail will have been rolled back. The rest of the rules are not tried once the first rule succeeds. This semantics is quite different from that of normal logic programming semantics, but has relevance to an operational semantics similar to what can be found in Prolog-type systems. The back-tracking through "recovery" micro-service is unique to iRODS and was needed so that a data system is not left in a corrupted or unstable state. Programmers who write micro-services should be careful to make sure that all actions are recoverable and should write corresponding recovery-micro-services. The rule-programmer can then use these to define rules. We presume that the rules will be written by scientists and those not well-versed in programming, but can work at a pragmatic level based on the semantics of the rules and micro-services.

More information on iRODS can be found here [16].

2.1 Complexity of iRODS

The implementation of a consistent, extensible, scalable, and evolvable data management system for an open repository requires integration of concepts from a wide variety of systems: data grids, relational databases, active databases and database triggers, logic programming and rule systems, server-side workflows, content management systems, and distributed operating systems. Data grids provide a means of accessing distributed storage resources using common interfaces with single sign-on for user authentication for access to data in diverse resources [30, 31, 32, 33]. Some data grids such as the SRB provide an integrated metadata catalog so that one can access replicated copies of files using logical names given to files. Others such as EUDataGrid and Globus Data Grid provide a tool kit where a user or community can put their own data grid together by integrating separate services. The SRB integrated system provides ease of installation, administration and usage and provides a uniform low-level API that is used to implement higher-order clients. The SRB has been used in multiple projects [34, 25, 35, 36, 37, 38, 39] and has been shown to handle Petabytes of data and 100s of millions of files. The logical naming paradigm and single sign-on authentication along with third-party authorization and metadata catalog services are the main ideas adapted from data grids into iRODS.

Relational databases [40, 41] play an important role in iRODS for the implementation of the integrated metadata catalog iCAT. In iRODS, as in the SRB, the metadata schema is quite complex. By using ANSI SQL conventions and 3rd Normal Form functional dependencies, the iRODS system implements an automatic query generation mechanism such that the user is exposed to a universal schema that hides the complexity of the underlying multi-table schema.

Rules and distributed rule execution are central to iRODS. The concepts for developing the rule language, the implementation of the rule engine and the transactional properties of rule execution rely heavily on concepts from active databases and database triggers, logic programming and rule systems and their semantics. The rules in iRODS are extensions to the Event-Condition-Action (ECA) rules/triggers of active databases [22]. The transactional property of each rule is maintained through recovery micro-services defined for each rule. This is an extension from the semantics of ECA rules in active databases. Triggers in databases use roll-back of database operations through database transactions. The semantics of the rule execution in iRODS are similar to that of operational semantics of logic programs (or their implementation as in Prolog). Indeed, the iRODS rule engine uses the backtracking mechanism of prolog-type languages so that when a rule fails, if there is another rule with the same name, then that is tried.

The iRODS rules control execution of a server-side workflow. This is in contrast to scientific workflow systems [42, 43] that execute the workflow at a compute server or at the client. The iXMS messaging system provides a simple way of distributed workflow data exchange. Unlike the Kepler workflow, where the director performs time-slicing of operations to enforce a semblance of parallel workflow operations, the system in iRODS provides true concurrent execution of the micro-services through the distributed rule engine.

Duraspace [20] and LOCKSS [21] are two systems based on digital library technologies. LOCKSS provides a persistent archive by managing multiple copies over the wide area network and is used for managing electronic publications across university libraries. Duraspace provides middleware for managing metadata and data for digital content. Multiple user interfaces have been integrated on top of Duraspace to manage accession services for ingesting data. Content management systems, like Documentum [44], Alfresco [45], Sharepoint [46], and Stellant [47] provide services needed to collaboratively create, edit, review, index, search, publish and archive digital files. These systems work within an enterprise and deliver a single common workflow package for managing large-scale and scalable data systems. A few of the systems also provide some form of rule-processing. Unlike iRODS, they do not use the concept of micro-services to create definable tasks that can be executed in a distributed chain to achieve a required goal.

The iRODS system, because of the aggregation of multiple technologies and paradigms, is unique and provides a platform for intelligent and evolvable open repository systems.

3. The iRODS Open Repository Framework

Open repository systems need an evolvable and scalable system for managing distributed data that may be shared by autonomous data providers and administrators. The iRODS system provides an ideal system for this implementation. Having presented the system description of iRODS, we show how it maps into the logical framework needed for an open repository framework.

The fundamental entity of the iRODS open repository framework is the concept of a *digital data collection* (or *collection* for short.) A collection is an aggregation of digital object that are "gathered together" because of some common logical characteristics. In iRODS, the collections form a hierarchical structure with collections having objects and sub-collections. The digital objects in a collection are physical objects that are located somewhere on the data network but are provided a unique identifier in the collection. The combination of the collection-hierarchy path identification and the name of the object in the collection together provide a unique *data object identifier* for each object in the open repository. Moreover, each iRODS open repository is given a name (called a zone name) that is unique and is registered in a zone authority [17]. The zone name and the unique object name in an iRODS collection provide a means for a global unique identifier (guid) [18] for each object registered in an iRODS system. The concepts of unique identification of digital objects and aggregation of objects into logical collections (for ease of browsing) are important features needed in an open repository framework. Moreover, these unique identification persist in the metadata catalog (iCAT) and provide a persistent identifier even if the object is physically moved from one repository to another.

A collection can span multiple administrative domains and storage locations – i.e., the objects in the collection can be owned, curated or administered by different people and agencies. The iRODS data grid organizes distributed data into a hierarchy of collections of objects that is independent of the location of the objects but provides a logical grouping that can be used to enforce uniform management policies across multiple administrative and storage domains – again a key need in a collaborative open repository. Policies for the open repositories are encoded as rules that govern the various operations that are allowed and performed at the collection-level. Even though the policies govern the life-cycle of a collection (and hence its component objects and sub-collections), there are other entities that also play an important role in an open repository framework. These include

1. User names, groups of users, resource names and resource pools- defining who are the users/owners/curators of collections and objects, and where the objects are located. These entities are also "uniquely" named inside an iRODS system independently of their network addresses and iRODS provides the necessary mapping.
2. Internal ontology that provides the schema of the system-wide metadata for the collections and other entities of the system (e.g. size and type of objects, role of a user (normal user, curator, etc), resource free space, etc)
3. User and domain specific (extensible) metadata system that captures non-systemic metadata needed for discovery and usage. These may include information about the object (e.g., telescope settings for an astronomical image), or process-centric information (e.g. flags needed by an ingestion workflow process),
4. A controlled vocabulary (or ontology) for defining and applying access controls and fundamental operations that can be performed on collections and objects,
5. Rule bases that encode the policies of the iRODS system and the micro-services (executable functions) that are the building blocks of the action part of the rules.

These entities are also abstractions that map physical names (such as data types and network addresses) to uniform logical names unique under the iRODS framework. This level of abstraction (or physical transparency) provides a means to design a system that is extensible and evolvable in time (the concept is similar to the data transparencies [19] provided by relational databases, enabling one to define schemas and queries without worrying about the internal implementation structures of the database). An advantage of such abstract name spaces in iRODS is that it becomes easier to migrate collections from one iRODS system to another as well from another data management system to iRODS and vice versa. The abstraction is also needed for long-term preservation – an important aspect of open repositories – as it enables evolution in system design, ontologies and implementations.

Our approach of using iRODS to define an open repository framework has several significant points of departure from the designs of digital libraries, portals, data grids and cloud storage systems, that makes it more suited for an open repository implementation. These include:

1. Explicit enumeration of the locations in the data management framework at which policy needs to be enforced. The iRODS system defines a minimal set of policy enforcement locations that enable the creation of generic infrastructure that can be used to support data sharing, data publication, data preservation, data analysis, and real-time data streams, by changing the management policies. Since an open repository system needs to be customized for each discipline that shares its data,

the underlying policies can be easily encoded using the iRODS system. Normally, in a portal like framework, policy management and repository management are kept separate with repository management being performed at the server-side and policies for ingestion, curation, sharing and long-term maintenance being done by explicit functionality encoded in the portal software. The outcome of a portal-based approach is that one can communicate with the repository only from the blessed portal entry-point and any other access would be completely disabled. In the approach taken by iRODS, by plugging in the policy at the server-side as part of the integrated repository framework, the client-side system is completely differentiated from the policy enforcement system. Hence, any type of client can be used (as appropriate for the community) without any penalty in policy enforcement. This again provides for multi-disciplinary use, as each discipline can choose clients appropriate for their data and usage model. Appendix A provides a list of "policy-enforcement" points in the iRODS system. When building an open repository using iRODS, the administrators and data owners can define checks and actions to be performed at these policy-enforcement points to customize the system for their needs.

2. Explicit enumeration of micro-services, the modules of executable code from which processing workflows can be composed. The need for policy enforcement at the repository server side requires one to provide the software functionalities that are needed for this purpose. In iRODS, we have defined a core set of such functionalities – called *micro-services* – that can be used in rules to encode the policies of the open repository. Micro-services can be viewed as well-defined software functions or procedures that perform a particular task. For example, iRODS has a micro-service for "replicateObject" which can be used to make copies of a digital object in two (geographically distant) storage resources and record the replication information in a metadata catalog. Users can discover and access the two copies under one logical object name. Another useful micro-service calculates a checksum for a digital object and stores that information in the metadata catalog. For the iRODS environment to be feasible, the level of composition of micro-services needs to be at a high level of granularity to simplify construction of procedures that enforce management policy. We list a subset of these micro-services that pertain to open repository management in Appendix B.
3. Explicit enumeration of the policies that are being enforced within the data grid. The policies that are needed by a community are encoded as iRODS rules which are very similar to the ECA rules found in active databases [22]. The policies are enforced through a distributed rule engine that is co-located with every storage system used within the data grid. Thus all operations applied by the data grid on its digital holdings are executed under the control of rules that are stored at the storage resource. This makes it possible to enforce management policies across administrative domains for retention, disposition, distribution, replication, time-dependent access control, integrity, authenticity, chain of custody, trustworthiness, Institutional Research Board access approval, HIPAA compliance, provenance.

4. Explicit enumeration of the types of structured information generated by the application of remote procedures. Since, the micro-services are fundamental blocks of policy enforcement, and they are chained together to form rules, intercommunication between micro-services is important. This requires a standard mechanism for inter-service communication. Similar to WSDL [23], that enables communication between web services [24], iRODS uses well-defined structured information for communication between micro-services. Since micro-services can be launched in multiple repositories that are distributed over a network, communication of the structure over the network is also necessary. For efficiency of local computation, the iRODS data grid had to be able to store in memory the structures generated by a micro-service, for efficient access by a chained micro-service. The structures also had to be linearized for transmission over the network to a micro-service at a remote location or to the client. This dual-nature of communication is enabled through the development of a mechanism to describe each structure, and pack and unpack the structures for transmission.
5. Explicit enumeration of the state information attributes required to implement a data management system. When micro-services operate, they require two types of information – information that is part of the immediate session and information that needs to be kept for long-term persistent after the session. The short-term session information is stored in the same structure that is used for inter-micro-service communication. For persistent state information, iRODS uses a metadata catalog (called iCAT) that stores persistent information in a relational database such as Postgres or Oracle or mySQL. The system provides a means to access the persistent information (say for a given data object) using a very simple query mechanism which is a subset of the SQL language. Also, each micro-service generates state information upon successful completion. The state information is saved to ensure consistent operation of the open repository. The state information constitutes the memory of the system, tracking the status of every record in the shared collection. This aspect is also important for open repositories because one needs to be able to consistently inform the users about changes being made to the system. By analyzing an audit trail, users and curators can track that proper operations were performed.
6. Explicit support for evolution of the data grid, through use of logical name spaces for first class objects that include users, files, resources, rules, micro-services, and state information. A major need was the ability to change a management policy and the associated procedures, and migrate data from the original collection which enforced the original policies, to a new collection managed by new policies. Through use of logical name spaces, versions of each first class object can be managed. Within the same data grid, the old rules controlling the old micro-services that generate the old state information for the original collection can be run in parallel with the new rules that control the new micro-services generating new state information on a new collection. A rule can control the migration of data from a collection controlled by the old policies to a collection controlled by new policies.
7. Support for deferred and periodic execution of rules to enable automation of the validation of assessment criteria. A rule can be written that checks whether the

current state information matches the desired values. If a discrepancy is found, such as a corrupted file caused by a disk head crash, the rule can access a valid copy and replace the corrupted file. Such checks need to be performed periodically since there are no perfect storage systems. Data may be lost through hardware malfunction, software malfunction, operator error, natural disasters, or malicious users. Assertions about properties of the shared collection are only as good as the set of assessment criteria that are used to validate the correctness of the system. Since policies can change over time, assessment criteria must also parse audit trails to determine the impact of policy changes. Given the ability to assess the system consistency, it is possible to have the system detect and repair problems, minimizing the amount of labor needed for an open repository administration.

3.1 Scalability and Performance

An important aspect of an open repository is its scalability – both in terms of the number of digital objects under its control and also in the ingestion rate and access characteristics under heavy load. The iRODS system has been shown to be highly scalable. The iRODS system used by the NARA TPAP project [25] has more than 15 million files and will have more than 100 million files in the near future (the project is aggregating EOS files from NASA Distributed Active Archive Centers) as an archived collection in the testbed. Experiments performed with iRODS have shown that it is capable of handling large file ingestions (50 files/second from a single stream), and degrades gracefully as the collection size increases. This result is reported in [26]. Several optimization techniques are also being tested and advocated for better performance of the iRODS system. These results give us confidence that an open repository implemented with iRODS can scale to 100s of millions of files and give good performance for ingestion and access.

4. Example Open Repository based on IRODS

The Temporal Dynamics of Learning Center (TDLC) [8] is one of six NSF Science of Learning Centers (SLC) [27]. TDLC aims to achieve integrated understanding of the role of time and timing in learning, across multiple scales, brain systems, and social systems. The scientific goal is therefore to understand the temporal dynamics of learning, and to apply this understanding to improve educational practice. Learning occurs at many levels: at the level of synapses and neurons; at the level of brain systems involved in memory and reward; at the level of complex motor behaviors; at the level of expertise learning; and finally, at the level of learning via social interactions between teachers and students. TDLC initiatives address such fundamental research questions as: How is temporal information about the world learned? How do the intrinsic temporal dynamic properties of brain cells and circuits facilitate and/or constrain learning? How can the temporal features of learning be used to enhance education? What are the best theoretical ways to conceive the temporal dynamics of learning in the brain and between brains?

Answering these questions cannot emerge from a single line of inquiry, so TDLC's research model is collaborative and interdisciplinary from the beginning. The center has created communities of scientists that cross disciplinary and institutional barriers in pursuit of these common research questions. Researchers in machine learning, psychology, cognitive science, neuroscience, molecular genetics, biophysics, mathematics, and education focus on these issues from multiple perspectives, synchronizing their research in parallel experiments in animals, people, and theoretical models. The center includes laboratories from 12 universities in the US, Canada, Australia, and UK. A significant challenge for collaborations among such geographically distributed scientists is sharing large quantities of data and stimuli quickly and easily, while carefully controlling access to only the collaborators permitted to view and manipulate the data.

One initiative of the TDLC is to develop and deploy innovative technologies to support this kind of data sharing in the learning sciences, not only for the TDLC but in also partnership with the other NSF Science of Learning Centers. The goal is to enable just-in-time sharing of neurophysiological data, motion-capture data, fMRI and electrophysiology data, and high-quality images and video across many laboratories. This requires easy, efficient, fault-tolerant transfer of hundreds of gigabytes, terabytes, and one day perhaps petabytes of data on a regular basis. Collaborators also need to be assured that shared data are seen only by those with permission dictated by human Institutional Research Board (IRB), HIPAA [28], and animal IACUC [29] protocols. And after a project or even TDLC ends, data need to be de-identified before sharing outside the immediate collaborative group, as dictated by IRB protocols. TDLC's challenge is technology for data sharing that includes speed, fault-tolerance, and sophisticated access control but at the same time is easy for scientists to install, maintain, and use on a regular basis.

The generality of the iRODS approach enables the implementation of TDLC specific policies. An example is the enforcement of Institutional Research Board approval flags for human subject data. The approval flags are managed in an independent institutional database that denotes the locations where human subject data may be distributed and the names of individuals that may access the data. iRODS rules are written that periodically harvest information from the administrative database, establish explicit access permissions for the named individuals, and set distribution approval flags for each file. At the policy enforcement point for data retrieval, all accesses are checked, verifying that both the distribution approval flag has been set and the access permission has been set for the specific individual. This approach makes it possible to independently control both access and distribution of controlled data.

6. Conclusion

We have discussed the requirements for an open repository framework. We have shown that the integrated Rule Oriented Data Systems (iRODS) is an effective candidate for implementing such an open repository system. We demonstrated this by first describing briefly the iRODS system and then showed how the salient features of that system make it a useful open repository system. Finally, we have shown an example open repository system under development that is being implemented by the TDLC community for sharing their distributed data. We have shown that their need for immediate sharing, discovery and processing as well as the need for long-term preservation for promoting new research and reuse of data objects are met by their iRODS based repository. Other groups that are implementing similar open repositories based on iRODS range from institutional repositories (the Carolina Digital Repository), to regional data grids (the Renaissance Computing Institute engagement center data grid), to preservation environments (the Duke Medical Archives), to data analysis systems (the MotifNetwork). An open repository is capable of supporting all types of data management applications.

Acknowledgement:
The research results in this paper were funded by the NARA supplement to NSF SCI 0438741, "Cyberinfrastructure; From Vision to Reality" - Transcontinental Persistent Archive Prototype (TPAP) (2005-2008) and by the NSF Office of Cyberinfrastructure OCI-0848296 grant, "NARA Transcontinental Persistent Archive Prototype", (2008-2012). The iRODS technology development has been funded by NSF ITR 0427196, "Constraint-based Knowledge Systems for Grids, Digital Libraries, and Persistent Archives" (2004-2007) and NSF SDCI 0721400, "SDCI Data Improvement: Data Grids for Community Driven Applications" (2007-2010).

References:

1. Australian Research Collaboration Service; Davis: A Generic Interface for SRB and iRODS, www.dhpc.adelaide.edu.au/reports/197/dhpc-197.pdf.
2. San Diego Supercomputer Center, http://www.sdsc.edu/.
3. Amazon Simple Storage Service (Amazon S3), https://s3.amazonaws.com/
4. Teragrid, http://www.teragrid.org/
5. US National Virtual Observatory, http://www.us-vo.org/
6. Ocean Observatories Initiative, http://www.oceanobservatories.org/spaces
7. IPlant Collaborative: Empowering a new plant biology, http:iplantcollaborative.org
8. Temporal Dynamics of Learning Center, http://tdlc.ucsd.edu/portal/
9. BIRN, Biomedical Informatics Research Network, http://www.nbirn.net/
10. Consortium of Universities for Advancement of Hydrologic Science, http://www.cuahsi.org/
11. DataONE: Enabling Data-Intensive Biological and Environmental Research through Cyberinfrastructure, http://mediabeast.ites.utk.edu/mediasite4/Viewer/?peid=38558e47202247bd847456b047cedfbd
12. Rajasekar, A., M. Wan, R. Moore, W. Schroeder, "A Prototype Rule-based Distributed Data Management System", HPDC workshop on "Next Generation Distributed Data Management", May 2006, Paris, France.
13. Data Intensive Cyber Environments Center, http://dice.unc.edu/
14. Aschenbrenner, Andres et. al., "The Future of Repositories? Patterns for Cross-Repository Architectures," D-Lib Magazine, November/December 2008, Volume 14 Number 11/1.
15. Rajasekar A, R. Moore, M. Wan and W. Schroeder, "Universal View and Open Policy: Paradigms for Collaboration in Data Grids" 2009 International Symposium on Collaborative Technologies and Systems, Baltimore, MD, May 18-22, 2009.
16. integrated Rule Oriented Data System, http://irods.diceresearch.org
17. SRB and iRODS zone authority, http://www.sdsc.edu/srb/index.php/Zone_Authority
18. Globally Unique Identifiers, http://www.ostyn.com/standards/docs/guids.htm
19. Codd, E.F. (1970), "A Relational Model of Data for Large Shared Data Banks", Communications of the ACM **13**(6): 377–387.

20. Duraspace open technologies for durable digital content, http://duraspace.org/index.php
21. LOCKSS Lots of Copies Keep Stuff Safe, http://www.lockss.org/lockss/Home
22. Dayal U., Active Database Systems: Triggers and Rules for Advanced Database Processing, Morgan Kaufmann Publishers Inc. 1994.
23. WSDL Web Service Definition Language, http://www.w3.org/TR/wsdl
24. Open Grid Services Architecture, http://*www.**ogf**.org/documents/GFD.80.pdf*
25. NARA Transcontinental Persistent Archive Prototype, http://www.dlib.org/dlib/july07/07inbrief.html
26. Rajasekar A, R. Moore, M. Wan and W. Schroeder, "Universal View and Open Policy: Paradigms for Collaboration in Data Grids" 2009 International Symposium on Collaborative Technologies and Systems, Baltimore, MD, May 18-22, 2009.
27. Science of Learning Centers, http://silccenter.org/nsf_slcs_index.html
28. Health Information Privacy, http://www.hhs.gov/ocr/hipaa/
29. Institutional Animal Care and Use Committee (IACUC), http://www.iacuc.org/aboutus.htm
30. Storage Resource Broker, http://srb.diceresearch.org
31. Globus Data Grid, http://www.globus.org/toolkit/docs/2.4/datagrid/
32. OGSA DAI, http://www.**ogsadai**.org.uk
33. European Union Data Grid, http://eu-datagrid.web.cern.ch/eu-datagrid/
34. Biomedical Information Research Network, http://www.nbirn.net
35. Data Grid Services Based on SRB for National Digital Archives Program in Taiwan Wei-Long Ueng, Hui-Min Lin, Eric Yan, In Storage Resource Broker Workshop, 2006.
36. SCEC: Southern California Earthquake Center, http://www.scec.org
37. ROADNet: Real-time Observatories Applications and Data Management Network, http://roadnet.ucsd.edu
38. NSDL: National Science Digital Library, http://www.nsdl.org
39. SIO Explorer, http://siox.sdsc.edu/
40. Introduction to Database Systems, Date, C.J., Addison Wesley, 8th Edition, 2003.
41. Database Systems: The Complete Book, Ullman, J., Garcia-Molina, H., and Widom, J., Prentice Hall, 2008.
42. Kepler, https://kepler-project.org/
43. Taverna, http://taverna.sourceforge.net/
44. EMC Documentum: Enterprise Content Management, http://www.documentum.com
45. Alfresco: Open Source Content Management System, http://www.alfresco.com
46. Microsoft Office Sharepoint, http://sharepoint.microsoft.com
47. Stellent, http://www.oracle.com/stellent/index.html

Appendix 1: Policy-Enforcement Points in iRODS

The policy-enforcement points in iRODS enable the open repository administrator, data provider and system curator to enforce customized operational checks and actions. Each open repository can have a different set of rules that are triggered at these points and lead to different behaviors for the system. We list the set of such points that are available in iRODS that are appropriate for open repository customization.

Each policy enforcement point has two sets of rules. The first set is applied before the action is performed and the second set is applied after the operation is performed. For example, the "On Creation of a Collection" point has two sets of rules, the first rule set is applied before the collection is created and the second rule is applied after the successful creation of the collection. In an open repository implementation, the administrator might have policies to be enforced both before (e.g. check collection-naming convention, check the role of the creator of the collection, etc.) and after (e.g. give access/write permission to groups of users, associate resources that can be used to store objects in the collection, put policy on how many replicas need to be created for each object ingested in the collection, etc.) the creation of a collection.

Collection-level Policy Points Policy-Enforcement Points:

1. On Creation of a Collection
2. On Deletion of a Collection
3. On Copying a Collection
4. On Moving a Collection
5. On Renaming a Collection
6. On Backing Up a Collection
7. On Versioning a Collection
8. On Replicating a Collection
9. On Listing a Collection
10. On Querying Metadata from a Collection
11. On Associating Metadata to a Collection
12. On Disassociating Metadata from a Collection
13. On Changing Metadata of a Collection
14. On Performing a discipline-centric operation on a Collection

Object-level Policy-Enforcement Points

1. On Ingesting an Object into a Collection
2. On Deleting an Object into a Collection
3. On Copying an Object from a Collection
4. On Moving an Object from a Collection
5. On Renaming an Object in a Collection
6. On Backing Up an Object in a Collection
7. On Versioning an Object in a Collection
8. On Replicating an Object in a Collection
9. On Querying Metadata of an Object in a Collection
10. On Associating Metadata to an Object in a Collection

11. On Disassociating Metadata from an Object in a Collection
12. On Changing Metadata of an Object in a Collection
13. On Performing a discipline-centric operation on an Object in a Collection

Appendix 2: Open-Repository-related Micro-services
Additional system-level micro-services useful for Open Repositories can also be applied as rules under the control the iRODS data grid. These micro-services implement specific functions required for discipline-specific applications, or advanced preservation functionality.

1. Compute and Register Checksum for an Object or Collection
2. Validate the Checksum of an Object or Collection
3. Recover an object on corruption from uncorrupted copies
4. Create a replica (copy) of an object in another storage resource
5. Maintain a record of all operations performed upon a file (audit trail)

Domain-specific micro-services useful for Open Repositories:
1. Extract metadata from an object using appropriate routines
2. Associate metadata to an object
3. Format Conversion routines (e.g. png to jpeg, MS Word to PDF, Open Office to PDF, ROM to netCDF, HDF to netCDF, etc)
4. Semantic Conversion routines (e.g. upload a selected subset of metadata from an XML file)

Assessment criteria:
1. Verify mandatory descriptive metadata attributes are present
2. Verify all records specified in a submission agreement are present in the collection
3. Verify that all Archival Information Packages conform to a specified template
4. Verify that all identifiers are unique
5. Verify all operations performed upon a record comply with current policy (requires processing audit trails)
6. Verify all accesses complied with stated policy