Policy-based Data Management

Reagan W. Moore,
Director DICE Center,
University of North Carolina at Chapel Hill
Arcot Rajasekar
Professor SILS
University of North Carolina at Chapel Hill
Wayne Schroeder
DICE Group Lead
University of California, San Diego
Michael Wan
iRODS Architect
University of California, San Diego

August 2011

Abstract

Data management applications, such as data sharing, data publication, and data preservation, manage different stages of the data life cycle. Each stage is characterized by a designated community, a concensus on the management policies, a concensus on the management procedures, and a concensus on the representation information. It is possible to virtualize the data life cycle by providing infrastructure that enables the evolution of the management policies, procedures, and representation information. Policy-based data management systems, such as the integrated Rule Oriented Data System, provide the ability to implement management policies, enforce the policies as computer actionable rules, and validate assessment criteria through queries on representation information or parsing of audit trails. The policies are applied independently of the choice of access client or storage repository, and can be modified to reflect new management requirements. Successful applications of policy-based data management are in production use for all stages of the data life cycle.

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, or the U.S. Government.

Introduction

The Data Intensive Cyber Environments (DICE) Group has been developing the integrated Rule Oriented Data System (iRODS) since 2006 with the goal of supporting all stages of the scientific data life cycle. (Rajasekar, Wan, Moore & Schroeder, 2006: Rajasekar, Moore, & Wan, 2010) Applications of the iRODS data grid range from formation of local collections for managing project data, to creation of a data grid for sharing data with collaborating researchers, to publication of data in digital libraries for access by the broader scientific community, to development of data processing pipelines to analyze entire collections, to creation of reference collections for long term preservation, to re-purposing of the original collection through federation with other data collections. In each application, the same iRODS infrastructure may be used to support the scientific data, but the policies and procedures that are applied to enforce management and control may be quite different.

The dominant lesson learned in the development of the integrated Rule Oriented Data System is that data management applications can be characterized by the abstraction levels of purpose, properties, policies, procedures, persistent state information, assessment criteria, and federation. A system that supports these abstraction levels (such as iRODS) can be tuned to manage each stage of the data life cycle. In particular, by selecting appropriate policies and procedures, a data management system can be created that implements the National Science Foundation data management and preservation requirements. The essential abstraction levels are:

- ***Purpose*** - reason a collection is assembled. The reason drives the choice of the types of digital records that will be included, the types of operations that will be performed upon the records, the context provided with each record, and the policies that will be enforced. Reasons for assembling a collection are typically tied to a specific stage of the scientific data life cycle. If the collection is being assembled for local use, then assumptions can be made about local knowledge and less context is required. If the collection is being assembled for use by a broader community, then much more complete contextual knowledge is needed to interpret the data correctly.
- ***Properties*** – assertions that will be made about the collection. The group assembling a collection typically will make assertions about completeness (collection contains all of the experimental data), or authoritativeness (collection contains data that come from a known source), or consistency (collection contains data that have been calibrated, registered onto a coordinate system, and converted to physical units). Desired properties might include the context required for each digital record (appropriate descriptive metadata), the data integrity (management of checksums and replicas), and chain of custody (audit trails of all operations performed upon the data).
- ***Policies*** - controls for enforcing desired ***properties***. Policies define when procedures are applied and the workflows associated with each procedure. It is not sufficient to apply policies only on ingestion and access. Policies are needed to govern all administrative tasks as well. This implies the need for policy enforcement points within the data management infrastructure that control all data management actions. The iRODS data grid manages 71 policy enforcement points, corresponding to actions such as deleting a user, ingesting a file, depositing metadata, deleting a file, adding a storage resource, selecting a storage resource

when ingesting files, etc. Policies are typically applied to control a desired action, or applied to do pre-processing before the action occurs, or applied to do post-processing after the desired action completes. The set of policy enforcement points is expected to grow slowly over time as new application communities decide that additional actions need to be controlled by a policy. During 2011, three policy enforcement points were added based on user requests.

- ***Procedures*** - functions that implement the ***policies***. The iRODS data grid chains basic functions (micro-services) together to implement workflows that are applied at the remote storage location under the control of a policy that is executed as a rule within a distributed rule engine. (Rajasekar, Wan, Moore & Schroeder, 2010) An implication is that the successful chaining of basic functions requires the exchange of structured information. The iRODS data grid manages the exchange of information through standard structures in memory, through parameter passing, through metadata in a catalog, through files, and through network communication. For operations that are executed across multiple storage locations, the in-memory structures are serialized and sent over the network to the remote location where they are unpacked and reconstituted as in-memory structures. This approach ensures that the workflows will work the same, whether they are executed entirely within a single storage platform or across multiple storage platforms. The iRODS data grid currently provides 250 micro-services. Note that about 67 standard in-memory data structures are sufficient to handle data interchange between the 250 micro-services.
- ***Persistent state information*** - results of applying the ***procedures***. The iRODS data grid manages 205 state information attributes about users, records, collections, storage systems, and rules. The attributes also include information about quotas, the load on the system, outstanding or deferred operations, audit trails, rule versions and rule execution.
- ***Assessment criteria*** - validation that the **state information** conforms to the desired ***properties*.** The assessment criteria can be evaluated through periodic rules that verify a well-defined property. This evaluation must directly address the issue of scale, as data grids may contain petabytes of data and hundreds of millions of files. Support for bulk operations is needed within the storage system, the metadata catalog, the rule engine, and the messaging system that is used to track progress.
- ***Federation*** - controlled sharing of **logical name spaces**. Given the wide range of failure modes, long-term data management requires replication across independently managed data grids. If a data management environment fails for whatever reason, an independently managed environment is needed to minimize risk of data loss. The sources of risk include media failure, software and hardware failures, operational errors, natural disasters, and security violations. Federation is also used to re-purpose the use of a collection, by integrating digital holdings across multiple sources under new policies and procedures.

Given this characterization of a data management system, we assert that the scientific data life cycle can be virtualized. This means that it is possible to manage the evolution of the policies and procedures that govern each stage of the data life cycle, while the purpose for the collection evolves. There is a strong correlation between the purpose for creating a collection and the stage of the data life cycle that is implemented. We observe that as the user community broadens, new stages of the data life cycle are entered. Thus data transitions from local use, to use in a collaboration, to use by the entire discipline, to use by future researchers, to re-use within a new

collection. Virtualization of the data life cycle corresponds to implementing the new policies and procedures that are needed to address the new objectives of the broader user community.

Lessons Learned

The implementation of data management infrastructure can be reduced to the creation of the policies and procedures that enforce the driving purpose behind the formation of a collection. A highly extensible data management system that supports addition of new policies, new procedures, and new state information can support all stages of the data life cycle.

There are multiple challenges in reaching this vision:

1. **Infrastructure independence.** The properties of the data management environment should be independent of the choice of storage and database technology. This implies that the same procedures should be executable on any operating system (Windows, Mac, Unix), and that the data should be storable in any type of storage system (file system, tape archive, cloud storage, hierarchical storage manager, social networking site). The iRODS data grid achieves this by having the micro-services issue an extension of the Posix-style I/O calls. The Posix I/O calls are mapped to the protocol of the underlying storage system by the iRODS framework. Separate storage drivers are written for each type of storage system. Note that 22 I/O calls are sufficient to handle all types of data management applications. For environments like cloud storage that are not capable of executing all 22 I/O calls, a compound resource is implemented. A disk cache on which partial I/O commands can be executed is associated with the cloud storage or tape system. Files are staged from the cloud storage system to the disk cache, where the policies are applied.

2. **Context.** The data management environment should maintain a context for each data record. In archives, the context is called representation information (provenance, descriptive metadata, data structure) and is used to interpret the meaning of the record. Representation information for the data management environment (policies, procedures, state information) is also needed to characterize how the data management environment is being controlled. An implication is that as policies evolve, the system must enable future managers of the collection to implement policies required for the future objectives. The context will evolve over time, and the changes to the context need to be tracked. In the iRODS data grid, this is achieved through instantiation of policies as computer actionable rules, and through managing versions of rules. In addition, audit trails of all operations can be generated.

3. **Validation of assessment criteria.** A data management environment is built from commodity components that are subject to multiple types of failure modes: media failure, in which files lose their integrity; hardware failure, in which disk heads crash and tape robots smash tape cartridges; software failure, in which software systems execute previously undetected bad code instructions under heavy load; operational failure, in which operators execute obsolete procedures that overwrite and lose data; natural disasters, in which fire or earthquake destroy entire archives; malicious users, in which vulnerabilities in the operating systems are exploited to

overwrite data. A data management environment can only make assertions about the last time that the assessment criteria were validated. The validations must be re-verified each time an assertion is made about the integrity of the data management environment. In the iRODS data grid, long-running processes are used to iterate through all of the files in a collection to verify a property. (Smith & Moore, 2006) The processes are executed in a distributed rule engine located at each storage system, guaranteeing that data only leaves a storage system after complying with the local policies.

4. **Persistent objects.** It is not sufficient to be able to incorporate new technology into the data management environment. The representation information for the data management environment contains policies and procedures that define the desired management plan. The management plan must be ported to each new version of the data management environment. In this view, the records are maintained in their original format. The procedures that parse the records are written using infrastructure independent representations of I/O commands, making it possible to port the procedures to new technology (operating systems). Since the original procedure can continue to parse and display the record, the dominant concern of format obsolescence is avoided. To enable use by new access methods, versions of the records will need to be created that are migrated to the formats needed by the more sophisticated display applications of the future. Thus the data management environment needs the ability to apply original procedures as well as future procedures to each record. This in turn may require the management of multiple versions of each record: the original format, and new encoding formats required by new display mechanisms. In practice, open source parsers for many data formats are being implemented as iRODS micro-services. This means that the data format never has to be considered obsolete. Instead the associated format processing procedure can continue to be applied into the future. (Moore, 2003)

5. **Data management as an active process.** At the point in time when new technology becomes available, both the old and new technologies are present. If data management is viewed as an active process, then incorporation of new technology is reduced to an interoperability problem. The new technology is integrated into the data management environment using the mechanisms that enable infrastructure independence. Policies can then be established for migrating records from the old technology to the new technology. For this approach to be successful, the data management administrator must continually seek to incorporate new technology. This has the advantage that new technology is typically cheaper, higher performing, and more sophisticated than old technology. An active data management process keeps a data management environment relevant by supporting access by the most recent display mechanisms. The iRODS data grid enables the incorporation of new technology by supporting multiple types of access methods, multiple types of authentication environments, multiple types of communication protocols, multiple types of storage systems, multiple types of operating systems, and multiple types of policy languages.

The design of the iRODS data grid was based on more than 10 years of experience with the Storage Resource Broker (SRB) data grid. (Baru, Moore, Rajasekar & Wan, 1998) The lessons learned in building and applying the SRB data grid were instrumental in the design of the iRODS data grid. Based on user experience with the SRB, the iRODS data grid manages policies as first class objects. This makes

it possible to apply different policies to users, files, collections, resources, storage systems, and any attribute stored as state information. The iRODS data grid can implement data management policies (retention, disposition, distribution, replication, access control, transformations, deletion control, etc.), automate administrative functions (migration of data to new storage systems, creation of usage reports), and validate assessment criteria (integrity checks, compliance with control policies, presence of required descriptive metadata). The ability to pick which set of policies should be enforced, create procedures by chaining micro-services, and validate assessment criteria make the iRODS data grid a very attractive platform on which to build a data management application.

Implementation

The lessons learned on implementation of iRODS are mainly concerned with:

1) System capabilities (whether the number of policies, procedures, and state information will remain bounded). Given the wide range of applications of the software, a major concern was whether the system would become un-maintainable or too complex to manage. This is related to:

- Number of policy enforcement points. There are 71 locations within the iRODS framework where policy can be automatically enforced. The number of enforcement points is expected to stay under 100, across all types of data management applications.
- Number of policies. The iRODS data grid associates policies with each policy enforcement point. In addition, time-dependent policies can be defined that are executed periodically. These typically validate assessment criteria, and automate execution of administrative functions. The number of policies can therefore grow larger than the number of policy enforcement points. The current release provides 83 default rules. In practice, we expect the number of policies to grow to the order of 200 rules. A new version of the rule engine has been developed to improve the ability of the system to handle large rules sets.
- Number of micro-services. The current iRODS release has 250 micro-services. We expect this number to grow to three hundred as format-specific procedures are created.
- Number of in-memory data structures. The current release has 67 standard data structures. This number is growing slowly to support new framework components.
- Number of I/O Posix calls. A total of 22 I/O calls are sufficient to support all current data management applications. This number may grow if additional calls are needed to support new types of applications.
- Number metadata attributes. About 205 state information attributes are used to manage the iRODS data grid. This number will increase as new micro-services are created.

The scale of the system framework can therefore be defined as 71 policy enforcement points, 83 policies, 250 micro-services, 67 standard data structures, 22 I/O calls, and 205 state information attributes. Systems that provide this level of support should be capable of implementing the full range of data management applications needed to support all stages of the data life cycle. A very interesting question is the definition of the minimal architecture that is capable of enforcing viable NSF data management and preservation policies. It is possible that a system that provides a smaller functionality set may be able to support NSF data policies,

but no attempt has been made to quantify the required set of abilities.

2) Scalability (how to manage hundreds of millions of attributes and petabytes of data)

- The number of user-defined metadata attributes can grow substantially. An example is the import of the NASA Moderate Resolution Imaging Spectroradiometer data set into an iRODS data grid, with 54 million files, 630 Terabytes of data, and more than three hundred million descriptive attributes.

  The major driver for metadata scalability is the choice of database technology, and the types of indexing needed to ensure that database interactions remain interactive. This has required additional database indices to keep the system running interactively for large numbers of directories. The iRODS data grid has been tested with 200 million files, and has maintained the required level of interactivity.

3) Single point of failure. For the iRODS data grid, concern is usually raised about using a central metadata catalog. In practice, this concern has proven to be unfounded. Three approaches have been implemented that remove concerns about a central metadata catalog:

   1. Creation of a high availability database server. Approaches include use of PGPool to distribute the central catalog across multiple servers, or use of an Oracle distributed database server. Both the Australian Research Collaboration Service and the French National Institute for Nuclear Physics and Particle Physics have implemented high availability database servers for iRODS data grids. Both systems are running successfully in production.
   2. Use of master-slave catalogs. The iRODS data grid supports slave catalogs that can be used for reads, with all writes directed to the master catalog. This allows reads to be done locally, while keeping writes synchronized.
   3. Federation of multiple data grids. Each data grid manages a separate metadata catalog. The catalogs are either periodically synchronized, or soft links are made between the data grids.

   The advantages of a central metadata catalog are that it is much easier to maintain consistency and support a single authoritative source that records all write transactions (a requirement for NIH data grids).

4) User requested functionality (how to implement the features required by users). A major support effort in management of data grids is the implementation of specific features requested by each new application domain. The effort expended in developing the technology can be broken down roughly as follows:

   - 35% for the design, development, testing, documentation, and bug fixes
   - 25% for porting clients to enable access by community specific interfaces
   - 25% for supporting application use of the technology (installation, consulting)
   - 15% for management and administration.

   Thus over half of the cost of developing data grids is spent on direct application

support (either new features, porting of clients, or consulting support). The types of new features that are requested include, for example, the ability to register a file as a replica of an existing file, exclusion of soft links when registering a collection into the data grid, and creation of utilities to manage encryption of data.

Support for community-specific clients is very important. Each application domain wants to preserve their original access method while accessing distributed data. Currently, more than 50 clients are capable of accessing an iRODS data grid. They include:

- Workflow systems (Kepler, Taverna)
- Digital libraries (DSpace, Fedora)
- Load libraries (Python, Fortran)
- Unix shell commands (iRODS iCommands)
- Grid tools (GridFTP)
- File systems (FUSE, WebDav)
- Web browsers
- Web services
- Windows browsers (iExplore)
- Dropbox (iDrop)
- Java based systems
- Community specific clients (VOSpace for astronomy)
- Portals (EnginFrame)

These interfaces are typically based on either Java (JARGON interface), Unix shell commands (iCommands), or a C library interface.

The iRODS data grid provides support for policy virtualization through the installation of a rule engine and rule base at each storage location. The policies for the data management environment are cast as rules. Each rule is defined for a specific event, governed by a condition, and executed as a workflow composed from basic operations called micro-services. Error recovery is executed through a recovery workflow. All requests for data at the storage system are processed through the rule engine. This ensures that the preservation policies are enforced independently of the choice of access client.

Specification of standard policy enforcement points

Most systems apply policies at the time of ingestion, and create a workflow that supports the validation of content, and storage/replication. However there are multiple additional places within a data management framework where policies should be enforced. Basically, every operation within a data management environment should be controlled by an explicit policy. Within the iRODS data grid data grid, the enforcements points correspond to application of policies before an operation is initiated, during the execution of the requested action, and after an operation is completed.

The question for interoperability standardization is whether these policy enforcement points are sufficient, or whether additional policy enforcement points are needed. This is equivalent to asking whether all of the generic operations that need to be executed by procedures have been trapped with pre- process and post-process policy enforcement points.

Table 1. Policies Invoked on File Ingestion

ApplyRule: acChkHostAccessControl
ApplyRule: acSetPublicUserPolicy
ApplyRule: acAclPolicy
ApplyRule: acSetRescSchemeForCreate
execMicroSrvc: msiSetDefaultResc(demoResc,null)
ApplyRule: acRescQuotaPolicy
execMicroSrvc: msiSetRescQuotaPolicy(off)
ApplyRule: acSetVaultPathPolicy
execMicroSrvc: msiSetGraftPathScheme(no,1)
ApplyRule: acPreProcForModifyDataObjMeta
ApplyRule: acPostProcForModifyDataObjMeta
ApplyRule: acPostProcForCreate
ApplyRule: acPostProcForPut
execMicroSrvc: msiSysReplDataObj(tgReplResc,null)

ApplyRule: acSetMultiReplPerResc
ApplyRule: acSetRescSchemeForCreate
execMicroSrvc: msiSetDefaultResc(demoResc,0)
ApplyRule: acPreprocForDataObjOpen
ApplyRule: acSetVaultPathPolicy
execMicroSrvc: msiSetGraftPathScheme(no,1)

Table 1 shows the enforcement of policies when a file is ingested into the data grid. A representative set of rules is shown. A full-fledged system will have more processing steps with additional collection-centric procedures. In this example, 10 policy points are applied when ingestion occurs. The first three policies are related to authentication and authorization. The second three policies are related to resource management (e.g. choice of resource based on attributes of the ingested object, or quota enforcement) and arrangement. The next two policies are concerned with system metadata extraction and/or assignment. The ninth policy in this list is concerned with the creation of the ingested digital object in the iRODS data grid and concerns all aspects for integrity and provenance (e.g. ownership) maintenance. The tenth policy provides a hook for post-processing of the ingested digital object as required by collection curator and/or the owner of the data object. The Post-process policy can be quite complicated, possibly distributed, and executed after a delay through a scheduling and queuing system. In this example, there are three policies for post-processing but only one is applicable (the other two had guard conditions that were not met by the properties of the digital object; the guard conditions can be based on combinations of type, arrangement, ownership, size, location, etc.). The one post-processing policy that was applied created a replica of the ingested digital object in another physical resource. This policy in turn invoked other policies as part of its execution. The post-processing steps are shown below the line within Table 1.

The example shows that even a simple process such as the ingestion of a file into a collection should invoke explicit policies that govern the desired properties. Each operation performed upon a data management environment should be controlled by an explicit policy that is migrated with the record. An administrator in the future should be able to verify the policies under which a record was ingested.

Specification of standard micro-services

Within a data grid, micro-services are chained together to implement each of the procedures required by the data management environment. The number of micro-services can be much larger than the number of assessment criteria. By composing selected standard micro-services into a workflow chain, it becomes possible to

implement a required management policy without having to write additional code. The number of micro-services currently supported within the iRODS data grid is on the order of 250.

The expression of procedures can be quantified by identifying the micro-services from which they are composed. Each micro-service should execute a well-defined standard function, such as file replication, or metadata extraction based on a template, or checksum validation, or query on state information. The standard operations within a data management environment can then be quantified as standard micro-services. The migration of policies to a new data management environment is simplified if the same standard operations are available within the new system.

A data management environment should be characterized by the policies that are enforced, the procedures that are executed, and the standard operations that are performed. In addition, each procedure generates standard state information, such as the location where the replica was made. It is possible to characterize a data management environment, and manage interoperability between systems through the ability to apply and enforce policies.

Conclusions

The iRODS data grid has been designed to support all stages of the data life cycle. By mapping from the purpose for building the collection, to required collection properties, policies, procedures, and state information, a data management application can be created that supports an explicit stage of the data life cycle. By managing evolution of the policies and procedures, a collection can be migrated to the next stage of the data life cycle. Hence within the same system it should be possible to provide NSF compliant data management plans that control both sharing of data, publication of data, preservation of data, and re-use of data. Based on the multiple applications of the iRODS data grid for each of these data life cycle stages, we strongly believe that the iRODS architecture is capable of serving as national data infrastructure. The lessons learned in implementing the technology strongly support the claim that policy-based data management systems can manage all stages of the data life cycle.

Acknowledgements

This research was conducted under funding from the National Science Foundation, Award Number 0910431, SDCI Data Improvement: Data Grids for Community Driven Applications, and the NARA supplement to NSF SCI 0438741.

References

[proceedings]Baru, C., Moore, R., Rajasekar, A. & Wan, M. (1998). The SDSC Storage Resource Broker. *CASCON'98 Conference. Toronto, Canada.* Toronto: IBM.

[journal article]Moore, R. (2003). The San Diego Project: Persistent Objects. *Archivi & Computer, Automazione E Beni Culturali, l'Archivio Storico Comunale di San Miniato.* Pisa, Italy.

[proceedings]Rajasekar, A., Wan, M., Moore, R. & Schroeder, W. (2006). A Prototype

Rule-based Distributed Data Management System. *HPDC workshop on "Next Generation Distributed Data Management". Paris, France:* Paris: IEEE.

[book]Rajasekar, A., Moore, R., Hou, C-Y., Christopher L., Marciano, R., de Torcy, A., Wan, M., Schroeder, W., Chen, S-Y. , Gilbert, L., Tooby, P. & Zhu, B. (2010). *iRODS Primer: integrated Rule-Oriented Data Systems*. San Rafael, CA: Morgan-Claypool Publishers.

[proceedings]Rajasekar, A., Wan, M., Moore, R. & Schroeder, W. (2010). Micro-services: A Service-oriented Paradigm for Scalable, Disributed Data Management. *24th IEEE International Parallel & Distributed Processing Symposium. Atalanta, GA.* Atlanta: IEEE.

[proceedings]Smith, M. & Moore, R. (2006). Digital Archive Policies and Trusted Digital Repositories. *2nd International Digital Curation Conference: Digital Data Curation in Practice. Glasgow, Scotland.*