Enabling Inter-repository Access Management between iRODS and Fedora

Bing Zhu
U. of California: San Diego
9500 Gilman Drive
La Jolla, CA 92093
(858)534-8373
bizhu@ucsd.du

Richard Marciano
U. of North Carolina at Chapel Hill
202 Manning Hall
Chapel Hill, NC 27599
(919) 962-0033
richard_marciano@unc.edu

Reagan Moore
U. of North Carolina at Chapel Hill
202 Manning Hall
Chapel Hill, NC 27599
(919) 445-9592
rwmoore@renci.org

ABSTRACT
Many digital repositories have been built using different technologies such as Fedora and the integrated Rule-Oriented Data System (iRODS). This paper analyzes both the Fedora and iRODS technologies to understand how to integrate the two systems to enable cross-repository data sharing. The areas considered include the digital object model, services, management of distributed storage, external data resources, and policy enforcement.

Categories and Subject Descriptors
H.3.7 [**Digital Libraries**]: Interoperability between Digital Repositories – object model, storage module, referencing external resources, policy management.

General Terms
Management, Design, Reliability, Verification.

Keywords
Digital Repository, Distributed Storage, Object Model, Digital Library, Metadata, Fedora, integrated Rule-Oriented Data System.

1. INTRODUCTION
Today, there are a variety of digital library technologies, including Fedora, DSpace, iRODS, EPrints, dLibra, and Greenstone Digital Library software [2, 3]. Multiple groups are exploring integration of these heterogeneous digital library systems [1, 16, and 17]. By enabling interoperability between repositories that are built from different technologies, we can promote data sharing. Our goal is to integrate the capabilities of multiple systems, allowing users to access different repositories via the tools with which they are familiar, while taking advantage of capabilities that are unique to a specific technology.

Early in the NSF National Science Digital Library (NSDL) project, an initial integration activity between the Storage Resource Broker data grid (SRB), and Fedora was developed. In this project, the SRB was used to manage a large repository of web materials retrieved from massive web crawls. The Fedora digital library middleware was used to build a portal that discovered and retrieved web pages stored in the SRB data grid. Several web services were developed by the SRB team so that the web archives could be accessed from the NSDL.

The integrated Rule-Oriented Data System (iRODS) [10, 15] is the second-generation middleware developed by the Data Intensive Cyber Environments (DICE) group at Univ. of North Carolina-Chapel Hill and Univ. of California, San Diego. It builds upon the concepts that were proven in the Storage Resource Broker [18] developed by the same group, providing a wide variety of functions to build and manage digital libraries, data grids, and distributed stores. The iRODS technology is being used by multiple projects, including the National Archives and Records Administration (NARA) and the French National Library to implement data management systems. As of today, the iRODS (including SRB) middleware supports very large distributed repositories that currently manage more than two hundred million files and petabytes of data [15].

The advantages of using iRODS to build digital repositories include:

- A uniform global naming space for managing digital datasets
- An efficient data access mechanism using parallel data transfer
- Access to data stored on distributed systems
- Rich interfaces including C/C++ API, Java API, Perl, Web Service, and Python
- Support for remote manipulation of data sets to minimize the amount of data sent over the network

The Flexible Extensible Digital Object Repository Architecture, or Fedora, provides a digital asset management architecture that can be used to build institutional repositories and digital archives [4, 5]. The integration of Fedora and iRODS can offer the following advantages.

- Fedora provides a rich presentation layer for objects, including digital objects and behavior objects.
- iRODS provides a distributed and efficient storage system.
- iRODS provides networked computational power to process data through micro-services, or workflows, inside clusters, UNIX boxes, and Windows machines.
- iRODS provides a suite of solutions for digital preservation of raw data such as data integrity verification, data replication, and disaster recovery.
- Fedora provides an explicit XML interface

2. Digital Object Model
iRODS and Fedora use different digital object models for managing digital assets. An analysis of the mapping between the two models indicates that both systems are capable of modeling digital objects that have complex structure.

2.1 iRODS Digital Object Model

An iRODS digital object, also called an iRODS dataset, contains digital content, system metadata and user-defined metadata. The digital content can be an entire file, or a subset of a file, or a query result from a database. iRODS objects are organized in a hierarchical collection-and-dataset architecture. Collections or sub-collections are virtually created in the iRODS catalog. Each dataset has a unique global UNIX-like path name. Physical data for an iRODS object is located in a designated, often remote, storage system that is specified by system metadata. Both system and user-defined metadata are stored in an iRODS catalog. The iRODS catalogs managed by different institutions can be linked together through federation.

2.2 Fedora Digital Object Model

The Fedora digital object model describes each digital object as a compound digital object, consisting of a persistent identifier (PID), an XML document for object property, and a number of datastreams [6] as is shown in the left side of Fig 1. A Fedora digital object is modeled in a FOXML XML schema [8]. Multiple digital entities within a single digital object are supported by using data streams. For example, a single web page usually contains multiple images and videos. When preserving this single web page, a virtual Fedora object can be created for this web page. The html content of the page, images, and videos can be saved as datastreams. Extra metadata can also be saved as a datastream in XML format.

2.3 Registering an iRODS Object into Fedora

Based on the above descriptions of the two digital object models, an iRODS file can be mapped or registered into Fedora through the following steps as depicted in Fig 1.

- Create a Fedora object. The full path of the iRODS object becomes the label of the Fedora object.
- Create an external reference datastream that points to the iRODS file.
- iRODS system metadata is extracted, wrapped into an XML document, and saved as a datastream in Fedora.
- iRODS user-defined metadata is extracted, wrapped into an XML document, and saved as a datastream in Fedora.

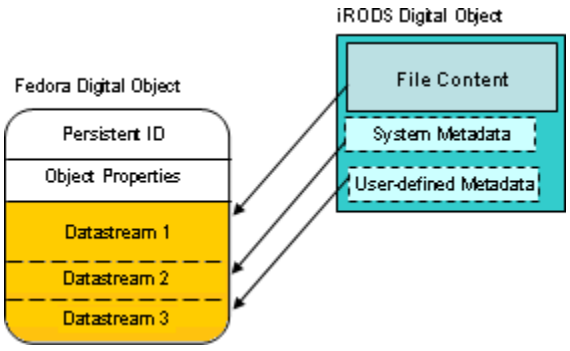

Fig 1. Registering an iRODS digital object into Fedora. The file content, system metadata, and user metadata of an iRODS digital object are mapped to three datastreams.

With this mapping, iRODS files can be registered into Fedora. Fedora provides the user with a front-end interface for managing iRODS files. Any change to a mapped object, including file content, system metadata, and user-defined metadata, must be synched back to iRODS. Besides the above approach which saves metadata as a datastream, it is possible for both system and user metadata datastreams to be registered as external datastreams through URLs. This approach will be described in section 4.2. A toolkit for bulk registration of iRODS files into Fedora is currently under development by the DICE team.

2.4 Modeling Fedora Objects in iRODS

An iRODS digital object is a single datastream or file. The iRODS team developed functions to manage structured files such as the Tar [19] format and HDF5 [9] format through corresponding iRODS drivers and micro-services. These functions provide I/O APIs to access individual files within a structured file. This provides iRODS the capability to contain multiple datastreams in a single dataset like Fedora. In storing an object, Fedora creates an FOXML object in a XML file and saves datastreams as individual files in the backend store. With a Tar file, all files for a Fedora object, including the FOXML file and datastreams can be glued into a single Tar file. When the Tar file is uploaded into iRODS with the "tar" object type, a user can request iRODS to expand the structure file into an iRODS collection. Thus each datastream or file within a Tar file can be individually accessed [11] and provides a similar naming mechanism for referring to an individual datastreams as that of Fedora.

One comment here is that we do not intend to use iRODS as a front-end for Fedora objects within this study. Our goal is to offer iRODS users a solution for modeling digital object with complex structure like that supported by Fedora.

3. Services

There are four types of object models in Fedora: data objects, service definition objects, service deployment objects, and content model objects. A service is basically defined by the service definition and service deployment. And the operation of the service on a particular digital entity is determined by its content model object. Since these objects are web service based and described in FOXML objects, their functionality can be encapsulated as iRODS micro-services. It is possible for iRODS to remotely manipulate FOXML objects. The Content Model service is a new version of the original disseminator model provided in Fedora 2.0. The model, however, is still a restricted service that applies only to data objects.

iRODS provides a rule-based micro-service model for executing workflows within the iRODS system [12]. In addition, iRODS services don't have the aforementioned restriction. A wide range of iRODS services has been developed for disk management, metadata extraction, and data replication. Currently, iRODS micro-services are pre-defined and compiled into the system. A sophisticated integration with Fedora can be realized through integration of Fedora services as iRODS micro-services.

4. iRODS Storage Module for Fedora

The iRODS data storage module for Fedora can be implemented in two ways, as a distributed data store to replace Fedora's default local store, or by referencing iRODS objects as external

datastreams within Fedora. These two iRODS storage interfaces provide a true distributed storage service for Fedora. The data storage module is implemented in Java on the Fedora server side by using Jargon [13], a Java toolkit developed by the DICE group. The implementation of the distributed data store is a follow-on to the DART project [7] at the University of Queensland, Australia. With this integration, iRODS can provide many functions for digital preservation such as periodic checking of data integrity, replication of data, and creation of deep archives [14].

4.1 iRODS serving as a backend store for Fedora

When using a local store, Fedora stores its objects (in XML) and datastreams separately in two directories, the data object directory and datastream directory. The iRODS storage module provides the Fedora administrator a choice beyond local storage by re-configuring a Fedora server to replace the local data store with an iRODS data grid. Fedora objects and datastreams can be distributed as depicted in Fig 2. In this approach, an iRODS user account is required to allow files to be saved and accessed in iRODS. When a Fedora object is created, it actually creates a FOXML object in iRODS.

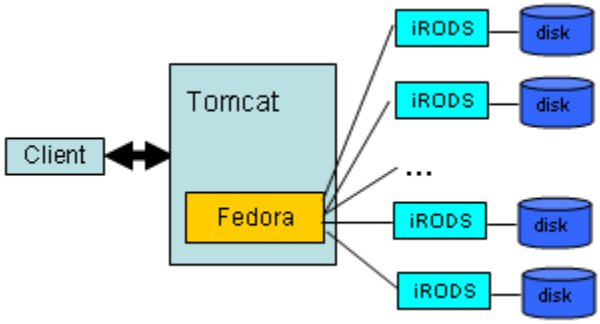

Fig 2. Fedora uses iRODS system as a backend Store. iRODS provides a distributed storage system for Fedora.

The iRODS storage module for Fedora is a standalone library that is independent of the official Fedora release and can be easily embedded into a Fedora server by re-configuring the Fedora storage module. The software can be downloaded from the iRODS web page at https://www.irods.org/index.php/Fedora.

4.2 Referencing iRODS Objects as Fedora External Datastreams

The Fedora object model supports multiple data streams inside a single object. Fedora implemented four types of data streams: internal XML metadata, managed content, external reference content, and redirect. Both external reference content and redirect types of datastreams use external data stored outside Fedora. The referenced object is currently a URL based object such as a HTTP file. An on-going effort aims to map each iRODS object as a recognizable external object through a URL with the following format:
irods://username:password@host:port/iRODS_file_path

Retrieving data with the above URL is already supported in the iRODS Jargon toolkit. This integration provides an extremely flexible mechanism for registering iRODS files into Fedora, in which iRODS files along with their system and user-defined metadata are virtually mapped as an iRODS digital object into Fedora as described in Section 2.3.

5. Policy-level Interoperability

Policy-level interoperability is a research area and is currently under development with the following research questions.

- Can a preservation environment be assembled from two existing repositories with differing management policies?
- Can the policies of the federation be enforced across both repositories, ensuring consistent management of the archives?
- Can policies be migrated between repositories, either by association of the policies with the storage repositories, or through control of repository procedures?
- What fundamental mechanisms are needed within a repository to implement new policies?

We are considering three basic integration embedding approaches (where **OM** stands for Object Model and **ra-DM** stands for rule-aware Distributed Model):

1. **ra-DM** driving **OM.** Design policy federation models that are implemented at the storage level through the ra-DM. A Fedora object model can be ported on top of iRODS, and rely upon iRODS to enforce the policies.
2. **OM** driving **ra-DM.** Design policy federation models in which the workflows within the OM model enforce the policies, but deposit the objects into the ra-DM. In this case, the policies enforced by Fedora control the management of the objects stored in iRODS.
3. **ra-DM** and **OM** co-driving. Design policy federation models in which policies are enforced by both types of preservation environments. This requires coordinating the impact of the policies between the systems, and exchanging state information that tracks the outcome of applying a management policy.

We will focus on the integration of an object model and a policy-aware distributed data model with Fedora and iRODS as representative software for each model. iRODS and its policy-aware repository of rules and built-in rule-engine will be used to implement and validate the integration concepts.

Summary

The study of the interoperability between iRODS and Fedora revealed that both systems are capable of modeling digital objects with complex structure. Both systems can be integrated so that data can be shared across repositories through virtual object registration, backend stores, and through external resource referencing. Policy-level integration will provide higher level integration of both iRODS and Fedora.

ACKNOWLEDGMENTS

We wish to acknowledge feedback from Will Owen, Dave Pcolar, Greg Jansen, and Steve Barr at the UNC Chapel Hill Libraries. They are prototyping an institutional repository based on integration of Fedora and iRODS as part of the Carolina Digital Repository (CDR) initiative. We are partnering with the CDR team and members of Research Computing (Willi Schulz and

Ruth Marinshaw) to explore the use of and integration with iRODS. In moving forward, a collaboration has been initiated with Fedora Commons principals (Sandy Payette, Dan Davis) to further explore policy-based interoperability mechanisms. The research results in this paper were funded by the NSF Office of Cyberinfrastructure OCI-0848296 grant, “NARA Transcontinental Persistent Archive Prototype”, (2008-2012) and by NSF SDCI 0721400, "SDCI Data Improvement: Data Grids for Community Driven Applications” (2007-2010).

REFERENCES

[1] Aschenbrenner A., et al. A Workshop Series for Grid/Repository Integration. D-Lib Magazine, Volume 15 Number 1/2, January/February 2009.

[2] Digital Library. http://en.wikipedia.org/wiki/Digital_library

[3] DSpace. http://www.dspace.org.

[4] Fedora. http://en.wikipedia.org/wiki/Fedora_Commons

[5] Fedora Commons. http://www.fedora-commons.org.

[6] Fedora Digital Object Model. http://fedora-commons.org/documentation/3.0b1/userdocs/digitalobjects/objectModel.html

[7] Fedora-SRB Database Integration Module. http://www.itee.uq.edu.au/~eresearch/projects/dart/outcomes/FedoraDB.php.

[8] Introduction to Fedora Object XML (FOXML). http://fedora-commons.org/documentation/3.0b1/userdocs/digitalobjects/introFOXML.html

[9] Hierarchical Data Format. http://www.hdfgroup.org/HDF5/index.html

[10] iRODS: Data Grids, Digital Libraries, Persistent Archives, and Real-time Data Systems. http://www.irods.org.

[11] iRODS Release Note 1.1. https://www.irods.org/index.php/Release_Notes_1.1

[12] iRODS Micro-Services. https://www.irods.org/index.php/Micro-Services.

[13] Jargon, a Java Client API for the DataGrid. https://www.irods.org/index.php/Jargon.

[14] Moore, R. Towards a Theory of Digital Preservation. The International Journal of Digital Curation. Issue 1, Volume 3, 2008.

[15] Moore, R., et al. Rule-Base Distributed Data Management Introduction. https://www.irods.org/pubs/DICE_Intro_to_iRODS-0806.pdf.

[16] Moore, R., Rajasekar, A., and Marciano, R. Implementing Trusted Digital Repositories. DigCCurr2007 International Symposium in Digital Curation. Chapel Hill, North Carolina, April, 2007.

[17] On the Need for a General Purpose Digital Object Repository. http://dltj.org/article/general-purpose-repository.

[18] Storage Resource Broker. http://www.sdsc.edu/srb/index.php/Main_Page.

[19] Tar File Format. http://en.wikipedia.org/wiki/Tar_(file_format).