A Service-Oriented Interface to the iRODS Data Grid

Nicola Venuti[*]*, Francesco Locunto*[*]*, Michael Conway*[**]*, Leesa Brieger*[◊]
[*]Nice S.r.l., [**]Data Intensive Cyber Environments Center, UNC, [◊]RENCI, UNC

Abstract

iRODS microservices and rules can be used to build a data grid that implements a community's own data policy. However, often the data administrators are not the developers who customize the services or deploy the data grid. A tool that gives the data administrator intuitive access to the rules and special-purpose services of their data grid is important in separating the IT tasks from the data administration tasks.

The EnginFrame (EF) cloud interface framework from Nice S.r.l. was used to build a service-oriented iRODS interface. This interface demonstrates how data grid access can be customized for community use; one view of the data grid, determined by data usage scenarios, is provided for the community user, and another view, determined by data management criteria, is provided for the administrative user.

Index Keyword Terms—iRODS data grid, data grid interface, data grid access, web interface, EnginFrame, EF, Grid Portal, cloud interface, administrative interface.

1. Introduction

Development of special-purpose microservices and rules will equip an iRODS data grid to implement specialized data access and preservation policy as required by a target community. The developers who would customize a data grid in this way may not, however, be the data administrators who determine and/or enforce data policy for that community.

Therefore, along with a customized data grid, it is imperative to offer a user-friendly interface that provides not only user access to community data, but also administrative access to the services that support and implement data policy. The data grid, with special-purpose services and with an administrative interface, then provides the data administrator with the necessary tools to curate and preserve his community's electronic data - without being an iRODS programmer to do it.

The user-friendly interface provides a separation between the data administrator and the systems administrator. It can offer intuitive access to the specialized data services, freeing up the data admin to concentrate on applying, enforcing, and verifying data policy for his community.

The authors used the EnginFrame (EF) cloud interface framework to develop a prototype of such an interface; this was used for a live demonstration of iRODS services at an NSF/NARA/NITRD iRODS presentation in August 2009. The interface was used to showcase important iRODS archival services in a real-time demo. It serves to illustrate how an interface can be customized to offer specialized views of the services implemented in a given data grid. Further, the interface presents one view of data and services for community users and another view, which includes more administrative functionalities, for the data administrator.

Several basic iRODS services were selected for the demonstration; we briefly mention implementation considerations for some of these special services, followed by a description of the EnginFrame interface and then the blending of the two technologies.

2. Specialized iRODS Services

While iRODS can be viewed as a framework for implementing data policy for the curation of electronic assets, it is also a tool kit that comes with many pre-defined rules, microservices, and capabilities. Some of these enable functionalities such as audit tracking and quota checking, in support of verification of policy; others enable capabilities such as searching on user-defined metadata.

These were the sorts of functionalities, based on out-of-the-box iRODS services, that were showcased at the NSF demo; thus these were the services exposed in the EF interface to the data grid.

2.1. Audit Tracking

Audit tracking is enabled in iRODS by changing the setting of the parameter auditEnabled from "0" to "2" in iRODS_root/server/icat/src/icatMidLevelRoutines.c, then recompiling, and restarting the iRODS server. Once audit tracking is enabled, any operation that calls upon the iCAT metadata catalogue is logged - in the iCAT. Any requests, such as downloading a data object, changing permissions on a collection, deleting or creating an object, etc., are all logged in the iCAT's audit table, along with record of the change that was made if authorization for the operation was granted. Audit information can then be tracked by querying this table and presenting the results in a user-friendly format. The queries can be implemented with the *iquest* icommand or with microservices by using msiMakeGenQuery and msiExecGenQuery.

There is a need to be careful, however, with these queries. The microservice queries use an iRODS-

specific syntax to approximate SQL but does not repli–cate it perfectly. Iquest allows a reduced form of SQL querying. Neither approach yet gives full SQL functionality. For audit table querying, there is a further complication that can result in spurious results. Consider that the audit table in the iCAT database contains the following fields:

```
AUDIT_OBJ_ID
AUDIT_USER_ID
AUDIT_ACTION_ID
AUDIT_COMMENT
AUDIT_CREATE_TIME
AUDIT_MODIFY_TIME
```

The audit table, in AUDIT_OBJ_ID, contains information about the entity (data object, collection, resource, user, etc.,) that is the object of an action that was performed and logged. It contains the ID of the target entity; however, there is no built-in mechanism to determine which it is - object, collection, user, resource, etc. Thus, at any one time, the AUDIT_OBJ_ID field of the audit table can refer to any of a number of tables containing detailed information on either a data object, a collection, a user, or a resource. The joins of the standard iRODS query services then have the effect of joining all the tables referred to by the ID, with the result that much spurious information is retrieved with the query.

By breaking down the joins into a series of simpler *iquest* queries, it is possible to separately query on each type of entity in the audit table, thereby avoiding the joins that cause spurious results to be generated. The following example for an audit procedure for an administrative user illustrates this; the *iquest* commands are run in a script so that output can be saved from one step to the next.

1.To see an audit trail for a given user, save an iRODS user name into a script variable and run the iquest *command to query the audit table:*

```
iquest "SELECT AUDIT_OBJ_ID,
    AUDIT_ACTION_ID, AUDIT_COMMENT,
    AUDIT_CREATE_TIME,
    AUDIT_MODIFY_TIME WHERE
    USER_NAME = '${_irods_username}'
```

2. Save the AUDIT_OBJ_ID into a script variable and use it to get query and get separate results from each entity table:

```
iquest "SELECT COLL_NAME, DATA_NAME
    WHERE DATA_ID = '${_objId}'"

iquest "SELECT COLL_NAME WHERE
    COLL_ID = '${_objId}'"

iquest "SELECT USER_NAME WHERE
    USER_ID = '${_objId}'"
```

For the NSF/NARA demo, the results of these queries were arranged into xml files to allow for formatted presentation. Additional Java filters provide an easy way to further manipulate the results and were applied in order to sort and refine the search results.

2.2. Other Services

iRODS allows users to add their own AVU triplets (attribute, value, units) to the iCAT metadata catalogue. Metadata searching of user-defined metadata was implemented for the demo using the *iquest* icommand to query the iCAT.

The implementation of quotas is awaited in iRODS and should be coming out in version 2.3. In the meantime, it is possible to use *iquest* to return and display usage information for each user, handling it similarly to the way quota information will be handled. This was implemented in the demo prototype.

The *irule* icommand allows users to run any iRODS rules on a command line. The interface also provided a means of pointing and clicking to edit and run selected rules.

3. EnginFrame

EnginFrame is proprietary software developed by Nice S.r.l. It is typically used as a computational grid portal or a cloud interface and serves as a framework for logically collecting applications, services and resources and presenting them in a web 2.0 interface that provides user-friendly access to the distributed resources. It is not a portlet container but instead delivers services that are JSR168-compliant; EnginFrame allows organizations to provide application-oriented computing and data services to both users (via Web browsers) and in-house or ISV applications (via SOAP/WSDL based Web services) so EF services could be used as portlets in another portal.

The main goal of EF is to hide the details and the complexity of the underlying infrastructure in order to improve usability and utilization. Usability goes up when end-user requirements for accessing the

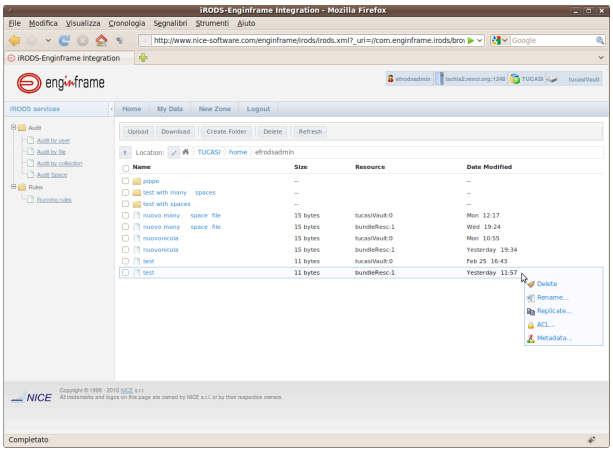

Figure 1. Metadata and ACL settings can be viewed and modified through the browser.

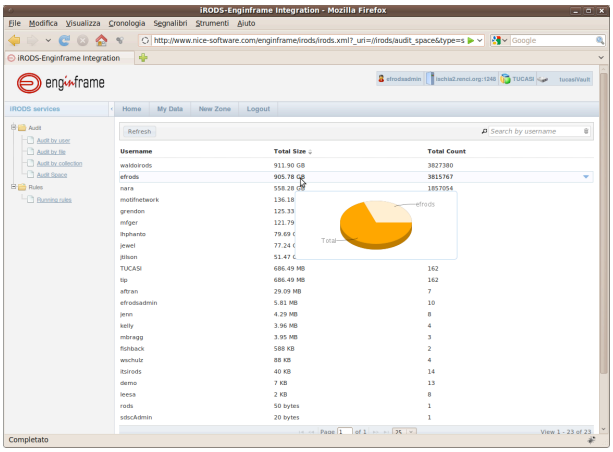

Figure 2. Usage data

infrastructure go down, and utilization is improved by making the evolution of the underlying systems trans–parent to the end-user and enforcing the utilization policies even as infrastructure evolves.

EF provides a flexible authentication framework with built-in support for a wide set of well-known authentication mechanisms like OS/NIS/PAM, LDAP, and Microsoft Active Directory. It has been integrated with the iRODS challenge-response authentication mechanism. The EF authorization framework allows the definition of groups of users and access control lists, thus providing a means for tailoring the Web interface to the specific users' roles or access rights. This was used in the demo interface to distinguish between community users and administrative users of the data grid. Community users were presented, in the interface, a reduced set of services compared to administrative users.

4. The iRODS EF Interface

The merging of the EnginFrame and iRODS technologies required development of an iRODS plug-in for EF and the wrapping of the iRODS services as EF services. The EF file manager for data browsing was also outfitted with iRODS functionalities so that some of the basic iRODS characteristics are present in the data browser.

User-defined metadata can be added, modified, queried, and deleted as part of basic iRODS functionalities. Setting and modifying ACL permissions are also included among the basic iRODS capabilities. Both these functionalities are available with the browser through the EF interface. See Figure 1.

Disk usage is queried using *iquest* and displayed.

The same sort of display is planned for quotas when that functionality becomes operational. See Figure 2.

Figure 3 shows the unfiltered results of an audit table query on all entries, and Figure 4 is a snapshot of the rule editor.

5. Deploying Data Grids

The customization of a data grid for a user community is an important step in deploying this technology for a given user group. Beyond simply installing the data grid, data management policy must be unambiguously defined and then translated into the microservices and rules of this technology.

Another very important step in the deployment is the development of a user-friendly interface for accessing the data grid. A custom interface can provide intuitive access to the custom services of the data grid and a user-friendly way of invoking the rules that implement and enforce data policy.

Further, the interface can be customized to various user groups that access the data and data services. As mentioned above, the EF interface was developed to show different views of the services to community and administrative users, thereby distinguishing between the different classes of services offered to the two groups. It would also be possible to adjust the view of the data grid to other user groups, so that the presentation of data and services fits with a group's own use cases.

6. The Future

A new domain of expertise will likely grow up around this technology, embodied by those who deploy the iRODS data grids. They will likely become increasingly separate from the DICE group who develops iRODS as well as from the user communities who are the consumers of the iRODS technology. There is in fact a need for a third group that bridges the gap between the other two. The developers know all that this

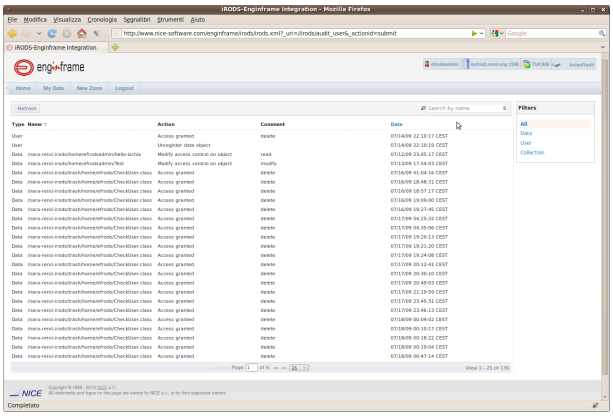

Figure 3. Data dump of the audit table

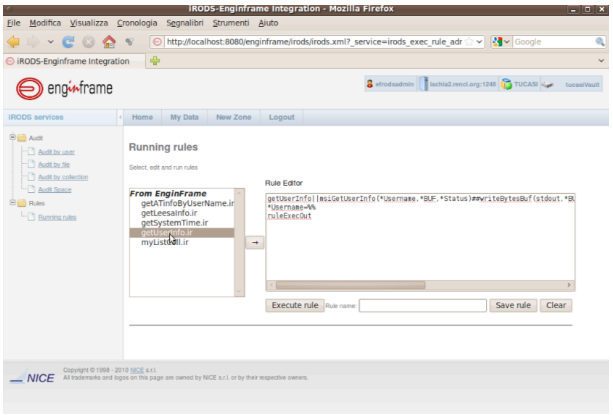

Figure 4. The rule editor

technology can offer, but are often not aware of the intricate details of the needs of the user groups. Users know some rudimentary aspects of the data grids but often define their needs in terms of the constraints they have learned to live with rather than exploiting the full potential of iRODS. There is increasingly a need for a group that straddles those two perspectives and brings rich iRODS capabilities to user groups with complex data needs.

These deployment groups must work closely with data specialists from the user communities in order to understand the required policy to implement in the data grids and how the administrative interfaces should operate. They will also have to understand how the users must view the data and services presented in order to meet their use cases. Policy should become easy to apply using the custom interface, and the full functionality of rich iRODS services should be delivered.

Deployment groups will promote the adoption of iRODS data grids, supporting communities who want to explore the technology, and allowing its adoption even by groups who may not be well-supported with in-house IT specialists. The deployment groups will do the programming of the services and the development of the interfaces so that users and data administrators will be freed from these tasks. The upshot is that many more communities will have access to this technology.

7. References

[1] Reagan W. Moore, Richard Marciano, Arcot Rajasekar, Antoine de Torcy, Chien-Yi Hou, Leesa Brieger, Jon Crabtree, Jewel Ward, Mason Chua, UNC Chapel Hill; Wayne Schroeder, Michael Wan, Sheau-Yen Chen, UCSD, "NITRD iRODS Demonstration", sponsored by NARA at NSF, 2009. *Can be linked from https://www.irods.org/index.php/Publications*.
"Technical Demonstration of Integrated Preservation Infrastructure Prototype", National Coordination Office for Information Technology Research and Development (NITRD) / NSF / NARA, National Science Foundation, Washington, D.C., August 4, 2009 Powerpoint Version. Combined Video and Powerpoint Slides of NITRD Demo. *Can be linked from https://www.irods.org/index.php/Publications*.
[2] iRODS and Data Preservation 2nd Workshop on Data Preservation and Long Term Analysis in HEP, Wayne Schroeder, SLAC National Accelerator Laboratory, Menlo Park, CA, May 26, 2009. *Can be linked from https://www.irods.org/index.php/Publications*.
[3] Policy-Based Distributed Data Management Systems, Open Repositories 09, Reagan Moore, Arcot Rajasekar, Mike Wan, May, 2009. *Can be linked from https://www.irods.org/index.php/Publications*.
[4] http://www.nice-software.com
[5] http://www.enginframe.com
[6] http://code.google.com/p/ef-irods-plugin/