The Development of a Native Cross-Platform iRODS GUI Client

Ilari Korhonen

June 11, 2015
Introduction

- Ilari Korhonen, working as a Systems Designer at the IT Services department, University of Jyväskylä, Finland
- Doing research IT infrastructure development at JYU
- JYU is a mid-sized university with about 15,000 students in 7 faculties and has a strong focus on research as well as education
iRODS at JYU

- My mission: a campus-wide iRODS data grid infrastructure for research data storage applicable to all fields of science
- Vast amount of requirements from different fields of science as well as legislation
- Physics, chemistry, biology, etc. produce large amounts of data in many different formats – both open and proprietary
- Social sciences, biology, psychology, etc. deal with sensitive data subject to legislation
- In almost all of the use cases proper metadata management is crucial
Research Data at JYU
Research Data at JYU
Research Data at JYU
Research Data at JYU
Research Data at JYU
Requirements for iRODS Deployment

- Secure data and metadata transfer
- Integration with external authentication (LDAP/Kerberos)
- Metadata extraction and management of some of the most crucial data formats in use
- Audit Trails for management of sensitive data
- High Availability and Scalability (no less than our EMC NAS)
- Ease of use – even for users with less technical skillsets
iRODS Clients

- iRODS has many different (kinds of) client applications
- The reference implementation being the iRODS icommands command line tools package
- The iDrop project at DICE has implemented a Java client and a web interface built on their Jargon Java iRODS library
- DICE has also lately implemented a WebDAV interface on top of Jargon to replace Davis – which is no longer supported.
- Also other projects have existed but are no longer being supported or even compatible with the current iRODS version
Project Kanki - Why?

- Goal: To build a fully native, cross-platform iRODS client application with a rich graphical user interface
- *Kanki* – *e.g. a rods in Finnish, cold or frost in Japanese*
- We really needed something to integrate seamlessly with iRODS 4.x to fully leverage the new modular architecture
- The ability to be able to use the iRODS 4.x auth and network transport modules out-of-the-box is great!
- Also, we did seem to have some special requirements for metadata management – for which we can now build custom metadata editors
The Benefits of a Desktop Client App

- Web applications still have a lot of limitations
- The numerous incompatibilities between different browsers – especially with the certain unmentionable one
- For example dealing with large iRODS data objects can be problematic because of memory issues
Why Go Native?

▶ Many reasons, one above all else – performance
▶ Also, seamless integration with iRODS 4.x features as well as the features of upcoming releases!
▶ E.g. Kerberos authentication and SSL transports work great
What About Portability?

- With native development portability issues are a reality
- This can be mitigated by using only std C++ and portable frameworks instead of OS interfaces
- A single codebase is ideal – which can be achieved
- The Qt framework has proven to be an excellent choice for cross-platform development
About Qt

▶ Originally developed by Haavard Nord and Eirik Chambe-Eng the two of which founded TrollTech, Inc. in Norway
▶ Stands for Q Toolkit – apparently Q was considered to be a pretty letter in Haavard Nord’s emacs font
▶ May 20, 1995 Qt 0.90 was uploaded to sunsite.unc.edu.
▶ Today Qt is actually Finnish owned and is the leading platform for cross-platform GUI development
▶ Many mobile and embedded platforms are supported as well
Some Points About Qt Development

- Qt heavily leverages threads so code should be thread-safe
- A thread safe calling convention called signal-slot interface
- To make the call interface easier, it is supported by extensive precompiler macros
- Qt 4 introduced a MVC (Model-View-Controller) architecture
- Abstract models can be extended to build custom models and associated with many different kinds of view objects (which Qt has many of)
- Also there is a UI compiler for building UI objects from XML
Project Kanki - So Far

- An object-oriented interface for iRODS
- Has all of the basic iRODS features implemented in the GUI
- A metadata editor with schema management with namespace separation and attribute management
- Compiles against iRODS 4.0 on both Linux and OS X (will do it with iRODS 4.1 next week)
- Windows support possible when it will be added to iRODS 4.x
- Still work in progress but soon to be released as beta
- A source release has been discussed and is probably out by the end of summer.
Kanki::RodsGenQuery metaQuery(this->conn);
int status = 0;
if (this->objDatum->objType == DATA_OBJ_T) {
    metaQuery.addQueryAttribute(COL_META_DATA_ATTR_NAME);
    metaQuery.addQueryAttribute(COL_META_DATA_ATTR_VALUE);
    metaQuery.addQueryAttribute(COL_META_DATA_ATTR_UNITS);
}
else if (this->objDatum->objType == COLL_OBJ_T) {
    metaQuery.addQueryAttribute(COL_META_COLL_ATTR_NAME);
    metaQuery.addQueryAttribute(COL_META_COLL_ATTR_VALUE);
    metaQuery.addQueryAttribute(COL_META_COLL_ATTR_UNITS);
}
// add a query condition for object name
metaQuery.addQueryCondition(this->objDatum->objType == DATA_OBJ_T ? COL_DATA_NAME : COL_COLL_NAME,
                              Kanki::RodsGenQuery::isEqual, this->objDatum->objName);
// if we are querying a data object also specify collection path
if (this->objDatum->objType == DATA_OBJ_T)
    metaQuery.addQueryCondition(COL_COLL_NAME, Kanki::RodsGenQuery::isEqual, this->objDatum->collPath);
// execute genquery and get status code from iRODS API
if ((status = metaQuery.execute()) < 0) {
    // error reporting code
}
else {
    std::vector<std::string> names = metaQuery.getResultSet(0);
    std::vector<std::string> values = metaQuery.getResultSet(1);
    std::vector<std::string> units = metaQuery.getResultSet(2);
}
To Do – Features To Be Added

- Full drag & drop integration to and from the desktop and inside the iRODS grid browser window
- A search interface with arbitrary criteria based on iRODS object attributes as well as AVU metadata
- Metadata validation against the configured schema
- Custom editors for metadata attributes
- A Rule Exec interface for submitting user rules to iRODS
- iRODS Access Control List Editor
- Synchronization of local directories to iRODS collections
- If you have suggestions?
Demo and Questions?

Contact Information:
▶ Ilari Korhonen, University of Jyväskylä (IT Services), Finland
▶ email: ilari.korhonen@jyu.fi

Thank you for your interest!