PROJECT APPLICATION

IRODS client for ImageJ

1. Executive summary

ImageJ is a public domain Java image processing program designed with an open architecture that provides extensibility via Java plugins. The aim of this project is to develop an ImageJ plugin which can be used to download/upload datasets from/to iRODS (Integrated Rule-Oriented Data System), an open source data management software used by the research community in order to take control of their data, regardless of where and on what device the data is stored.

2. The Project

2.1 How are you going to implement it?

My plan for this project is the next one:

- *Setup a simple iRODS infrastructure using a Virtual Box machine*

Following the steps described on the iRODS blog, I will setup the catalog (I-CAT iRODS) and the resource in a single virtual machine. My work can be verified using this iRODS testbed. This will ease the communication with the mentors and helps us to avoid problems related to different testbed setups.

- *Develop the functionality for download/upload from/to iRODS*

The starting point for my plugin will be the existing Dropbox plugin for ImageJ. Using the existing GUI , I will focus on the effective communication between ImageJ and iRODS which will be done using the Jargon API.

- *Integrate the iRODS plugin with Dropbox plugin*

Both iRODS and Dropbox are related with the cloud architecture so we should design a single plugin which integrates both functionality. The only **difference** will be **at the authentication step** where the user will be asked whether he wants to connect to iRODS or Dropbox. The file manager will be the same, both for iRODS and for Dropbox.

- *Improve the existing GUI*

I believe that the file manager from the Dropbox plugin can be improved by enhancing the ease of use and reducing the time required to download a series of files. My plan is to redesign the file manager with a Drag and Drop functionality. Also, I plan to use progress bars in order to inform the user about the progress of the download/upload instead of using messages. In order to easily build a modern user interface I will use Window Builder. Window Builder is composed of SWT Designer and Swing Designer and makes it very easy to create modern Java GUI applications.

2.2 Detailed description of the approach

- *Develop the functionality for download/upload from/to iRODS*

The first step is to setup Eclipse IDE in order to run an ImageJ plugin following the steps from ImageJ Docu. Using these steps, I will run in Eclipse the existing plugin written for Dropbox which will be used as a starting point.

The next step is to import the Jargon project into my iRODS project. The Jargon project is composed of a series of smaller projects: jargon-conveyor, jargon-data-utils, jargon-httpstream, jargon-ruleservice, jargon-ticket and others The API for transferring data can be found in the jargon-core project. I will create a jar composed of the jargon-core functionality using the steps from Jargon Wiki. Shortly, I will run a maven build which generates the jargon-core-4.0.2.1-SNAPSHOT-jar-with-dependencies.jar jar. This jar will be used by my iRODS plugin.

In order to do the authentication to the iRODS server, I will use I will use the IRODSAccount class from org.irods.jargon.core.connection in order to specify:

- ✓ iRODS user
- ✓ iRODS Password
- ✓ iRODS Zone
- ✓ default user resource
- ✓ IP address of the iRODS server (in testing phase, this will be the IP of the Virtual Box machine)
- ✓ listening port for the iRODS server (the iRODS service listens on port 1247 by default)

I will use the IRODSSession class from org.irods.jargon.core.connection in order to start a session with the server using the above account. I could easily integrate the new authentication method to iRODS in the existing Dropbox plugin by using a different action listener for “Access Dropbox” button.

The iRODS filesystem will be accessed using classes from the package org.irods.jargon.core.pub.ro. The operations from the IRODSFileFactory interface allow an easy manipulation of the files from the IRODS server.

The process of uploading a file is composed of two steps: creating the file remote, then transferring the content of the local file to the remote file. Creating a file on a remote iRODS server can be done using the method instanceIRODSFile exposed by IRODSFileFactory. This method creates an instance of an IRODS file at the path specified as a parameter. In order to transfer a local file to the remote iRODS server, I will use the java.nio package for storing the local file in a byte array which can be passed as a parameter to a stream of type IRODSFileOutputStream which manages the actual transfer.

A sample code for uploading a file is in the next chunk of code:

```
IRODSFile irodsFile = irodsFileFactory.instanceIRODSFile(TargetiRODSPath);
irodsFileOutputStream = irodsFileFactory.instanceIRODSFileOutputStream(irodsFile);
File fi = new File(FileLocalPath);
byte[] fileContent = Files.readAllBytes(fi.toPath());
irodsFileOutputStream.write(fileContent, 0, fileContent.length);
```

I could easily integrate the new upload method in the existing Dropbox plugin by rewriting the DbxUploadFIle from DbxUtility class.

The process of downloading a file is composed of two steps: creating a file locally followed by the transfer of the content of remote file to the local file. The key in downloading a remote iRODS file is obtaining an irodsFileInputStream handler for reading the contents of the file in a byte array. After that, we can write this byte array to a local file. It's worth mentioning that reading/writing files as byte arrays makes the process of downloading/uploading very general because it works for every type of file, whether it is a .png image or .odt document.
I could easily integrate the new download method in the existing Dropbox plugin by rewriting the DbxDownloadFIle from DbxUtility class. A sample code for uploading a file is in the next chunk of code:

```
IRODSFile irodsFile = irodsFileFactory.instanceIRODSFile(FileiRODSPath);
IRODSFileInputStream irodsFileInputStream =
irodsFileFactory.instanceIRODSFileInputStream(irodsFile);
byte[] b = new byte[(int)irodsFile.length()];
irodsFileInputStream.read(b, 0, (int)irodsFile.length());
fos = new FileOutputStream(TargetLocalPath);
fos.write(b);
```

In order to upload a dataset (for example a directory) I will have to create directories in the remote iRODS server. Jargon API offers the possibility to create remote directories, using the mkdir method exposed by IRODSFile interface:

```
IRODSFile irodsFile = irodsFileFactory.instanceIRODSFile(TargetDbxPath);
irodsFile.mkdir();
```

I could easily integrate the new upload method in the existing Dropbox plugin by rewriting the DbxUploadFolder from DbxUtility class.

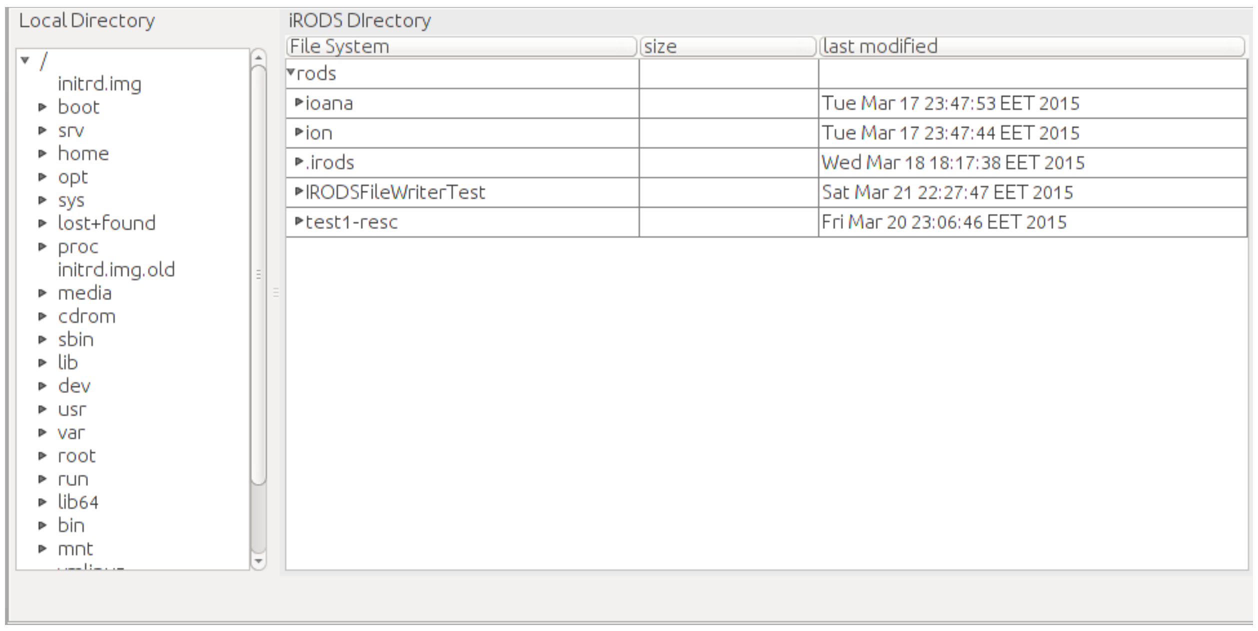
- *Integrate the iRODS plugin with Dropbox plugin*

I want to use the same plugin, both for accessing the Dropbox and the iRODS files. The difference will be at the authentication step where the user will be asked if he wants to connect to iRODS or Dropbox account. Specifically, I will add to the left JPanel of the mainFrame two new radio buttons with the options: “Login Dropbox”or “Login iRODS”. Depending on the user selection, the left JPanel will contain the fields necessary for Dropbox connection or for Irods connection.

- *Improve the existing GUI*

In the existing implementation, if the user wants to download a file, the following steps are required:
- select the radio button for “Download”;
- press the “Select” button in order to open the file manager for the source file;
- every time the user wants to go further in the file tree, push the “Expand” button;
- when the user decided the right download file, select the “Select” buttton;
- the last three steps are reiterated in order to select the destination path.
- press the “Start” button.

This can be time-consuming if the user wants to downloads multiple files from different places. My proposal is to redesign this GUI with a Drag and Drop interface. My plan is to create two JTree data structures, one corresponding to the local file system and one corresponding to the remote file system and implement the functionality for drag and drop between this two JTree, something similar to the iDrop-Web client for desktop:



Local Directory

```
▾ /
    initrd.img
  ▸ boot
  ▸ srv
  ▸ home
  ▸ opt
  ▸ sys
  ▸ lost+found
  ▸ proc
    initrd.img.old
  ▸ media
  ▸ cdrom
  ▸ sbin
  ▸ lib
  ▸ dev
  ▸ usr
  ▸ var
  ▸ root
  ▸ run
  ▸ lib64
  ▸ bin
  ▸ mnt
```

iRODS DIrectory

File System	size	last modified
▾rods		
▸ioana		Tue Mar 17 23:47:53 EET 2015
▸ion		Tue Mar 17 23:47:44 EET 2015
▸.irods		Wed Mar 18 18:17:38 EET 2015
▸IRODSFileWriterTest		Sat Mar 21 22:27:47 EET 2015
▸test1-resc		Fri Mar 20 23:06:46 EET 2015

Another functionality that I want to add is the file transfers bars for showing upload/download progress. This could be done using the JProgressBar component from Java Swing.

3. Implementation

3.1 Minimal set of deliverables

Deliverable 1: Plugin with a simple GUI, asking for iRODS credentials. The plugin can create a session with the Irods server and display a successful/not successful connection message.
Deliverable 2: Plugin wih the capability to browse the local file system and the remote iRODS file system.
Deliverable 3: Plugin with upload and download file functionality from iRODS.
Deliverable 4: Plugin with upload and download folder (trees) functionality for iRODS.
Deliverable 5: Plugin with Dropbox functionality integration
Deliverable 6: Plugin with Drag and Drop functionality.
Deliverable 7: Plugin with Drag and Drop and Progress Bar functionality.
Deliverable 8: Public Documentation, something similar to: http://atin007.github.io/dbclient/

3.2 Timeline / tasks deliverables

Week 1: Deliverable 1
Week 2: Deliverable 2
Week 3-4: Deliverable 3
Week 5-6: Deliverable 4
Week 7-8: Deliverable 5
Week 9-10: Deliverable 6
Week 11: Deliverable 7
Week 12: Deliverable 8

3.3 How will you communicate with the mentoring organization?

I will be available every day on Gmail and Google Hangouts: gucea.doru@gmail.com
I also noticed that the mentors have Gmail addresses so this communication method is familiar both for me and for my mentors.

3.4 Are you interested to further support the project after the end of the fellowship?

Definitely. The Jargon API is in an early stage of development and I plan to keep this plugin updated with the latest version of Jargon API such that the ImageJ users will always have a great User-Experience. This is also the aim of Deliverable 8 where I will keep a log with all the modifications and I will keep in touch with the users of the plugin.

4. The candidate

4.1 What is your motivation for the specific project?

I applied to this project because I believe the result of my work would ease the work of researchers who want to easily manage their ImageJ data in the cloud. Also, I am aware that most of the data will move in the cloud in the future and I want to build a deeper understanding of the cloud architecture.

4.2 Is this the only project that your are interested in?

Yes, I liked this project idea ever since I read the project proposal and this is the only project that I applied for.

4.3 What makes you a good candidate for the project?

First of all, I used Java a lot both at the university and in one of my summer internships. Secondly, I am also an active contributor to another open-source project, MPTCP (http://www.multipath-tcp.org/), so I know how to communicate with others members in order to express my ideas and offer/get help.

4.4 Experience

a. Do you have experience in Java

I used Java in one of my summer internships where I developed a module for Internet banking. I written both the front-end (Java Swing, Java Server Faces) and the back-end of the module (JSON, REST services).On the other hand, I used Java for about 4 years in my homework assignments from college.

b. Do you have experience in ImageJ

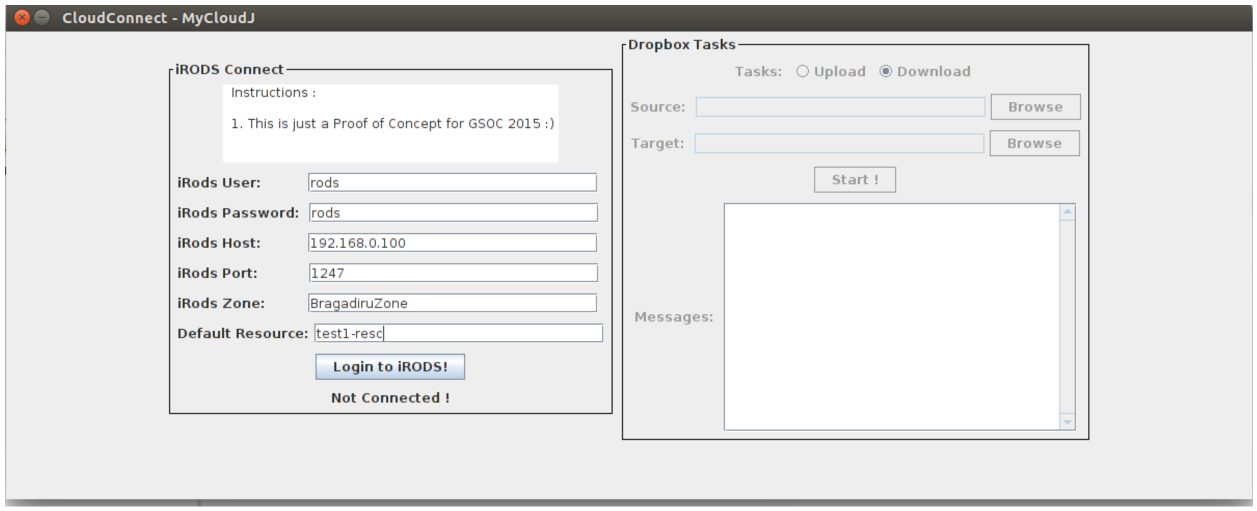
I have about 90 hours of experience writing a demo plugin for this project. The iRODS plugin for ImageJ can be found at:
https://drive.google.com/file/d/0B5SBH08PU_ChNTVGYjRJRTNfa0U/view?usp=sharing

My plugin also requires another jar to be placed in the plugin folder of ImageJ:
https://drive.google.com/file/d/0B5SBH08PU_ChbWVET29vN2Z5c2s/view?usp=sharing

I also setup a testbed for testing this plugin using a Virtual Box machine. The password for account doru from this Virtual machine is incf. The machine can be downloaded from:
https://drive.google.com/file/d/0B5SBH08PU_ChV3A2cXNYaS1fdjg/view?usp=sharing

Instructions for running the demo plugin:

- download the virtual machine, untar it and start the Virtual machine using VirtualBox. You will have to install Virtual Box on your physical machine (sudo apt-get install virtualbox). Get the IP of the eth0 interface from the VM using ipconfig.
- copy the the above two jars in the plugin directory of ImageJ
- Start ImageJ and access MyCloud plugin from the Plugins Menu. This screen should be displayed:



CloudConnect - MyCloudJ

iRODS Connect

Instructions :

1. This is just a Proof of Concept for GSOC 2015 :)

iRods User:	rods
iRods Password:	rods
iRods Host:	192.168.0.100
iRods Port:	1247
iRods Zone:	BragadiruZone
Default Resource:	test1-resc

Not Connected !

Dropbox Tasks

Tasks: Upload [✓ Download]

The Virtual Box machine is configured with two users:

- user rods with password rods
- user test1 with password test

The iRODSZone is BragadiruZone and the the default resource is test1-resc. The iRODSHost is the IP of the VirtualBox machine and the listening port for iRODS is 1247.

After authentication, you can use the plugin for upload/download files/folder to/from iRODS in the same way you did with the DropBox plugin.

4.5 Will you be working full time?

From my estimation I will work about 30 hours/week at this project. I already invested a lot of time in my demo plugin so I can invest more/less time in this project depending on the problems encountered. However, I learned an important lesson: in software, the success of a project is not measured in hours :).

4.6 CV