DMLite plug-in for Rucio

written by
Daan van Dongen

under the supervision of
Mario Lassnig

September 2013

Introduction

In the beginning of April I got accepted as a summer student to work on a project of the Rucio data management system for a few months, starting in the beginning of July. I did this project under the supervision of Mario Lassnig.

The ATLAS project Rucio is a data management system and is specifically designed to handle the large amounts of data used by the ATLAS collaboration. As a summer student I was assigned to develop an extension to Rucio that allows users of the system to access the data and the structure in the namespace via web browsers and Linux shells.

The Plug-in

DMLite is a library designed to handle communication between an Apache web server and a data storage system. Using this we can set up an Apache server, which can be accessed via the HTTP/WebDav protocol, for instance, via a web browser. A DMLite-plugin can then be written to gather information from a data storage system. The plugin I wrote translates a request sent to the Apache server to the corresponding request to Rucio, which is sent via the JSON communication protocol over HTTP. The response from Rucio is then interpreted by the plugin and the requested information is presented to the user. The plug-in is mostly written in C++.

Another way to acces the plug-in next to web browsers, is davfs. This allows access to an Apache server as though it is on a local disk. This way we can use Linux shell commands to access the Rucio file system and browse and manipulate the data like POSIX.

The two main functions that needed to be implemented in the DMLite plug-in were, in first place, correctly translating the Rucio data structure to a directory structure and, in second place, generating and sending metalink files of data in the system. The first function allows users of the plug-in to browse through the available data in the Rucio file system, as shown in Figure 1. The second function allows users of the plugin to indirectly access the data. The data objects themself aren't stored on the Rucio server, merely a list of servers with replicas of the data is. The implemention lets the user download a metalink file for a data object, which will redirect the user to a currently available replica of the requested data.

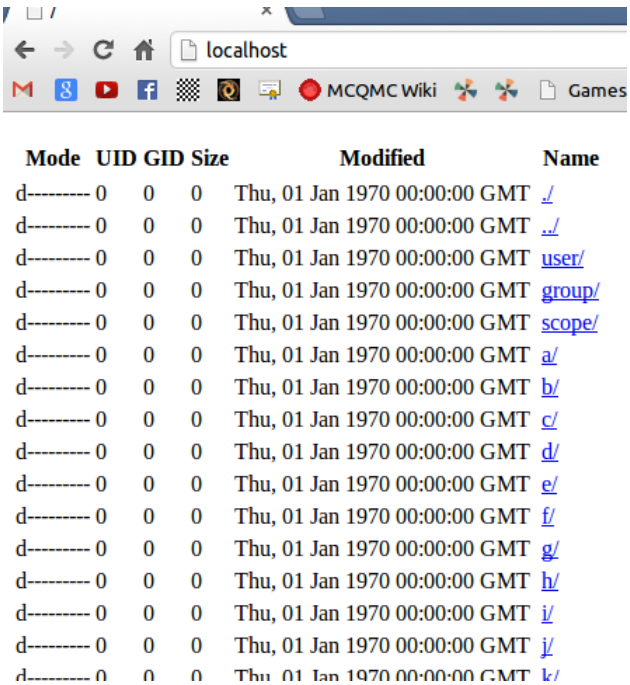

Figure 1: *The first page presented when browsing through the database in a web browser.*

One of the main issues I came across was related to the fact that web browsers and Linux shells use different methods of the same protocol to access data, which often made possible implementations only work for merely one of these two cases. This was, for instance, the case when trying to access a subdirectory: A web browser would send a request containing both the previous directory and the directory it would like to open, which are both needed for easy access to the corresponding data in Rucio. A Linux shell, however, misses the information about its current directory in its request. These issues were mostly solved by applying an implementating that worked for both cases. For the given example, this was solved adding the information of the current directory in the name of the directory that the user likes to access. There have been exceptions were no implementation could be found that worked for both methods. In those cases, different solutions were implemented for both methods.

Another main issue was finding a solution to the conflict between DMLite's built-in function to generate and send metalink files, which uses redirection, and Linux shells, that do not allow redirection. The solution to this was handling the request for a metalink file within the Apache configuration and reimplementing the metalink functionality within the DMLite library.

The third main issue was dealing with the large databases in general. Rucio is designed for large-scale operations on large data structures. However, most shell functions are not made to deal with these large databases and will often flood the server with requests. For instance, when Bash, a Linux shell, checks if a given path is a directory, it will try to open that directory and try to read it contents. If this operation doesn't fail, it gets listed as a directory. Normally, on a local system, this is checked within a moments notice, but when dealing with thousands of directories, that all need to be checked by Bash, each containing thousands of items, the time it takes quickly ramps up. Especially since these directories can only be accessed by sending a request to the server. This was solved by reimplenting some of the troublesome Bash functions.

The resulting plug-in, together with a script that reimplements some Bash functions, is able to quickly and correctly translate the data stored in Rucio to an intuitive directory structure for both web browsers and Bash shells. It also allows the downloading of individual metalink files in both cases. The reimplementation of the copy command in the Bash script allows the user of the shell to download metalink files for all files in a directory at the rate of about 1 per second. When a reference is given to this function, it will even send only metalink files of files in the directory that contain that reference within their name, as is shown in Figure 2.

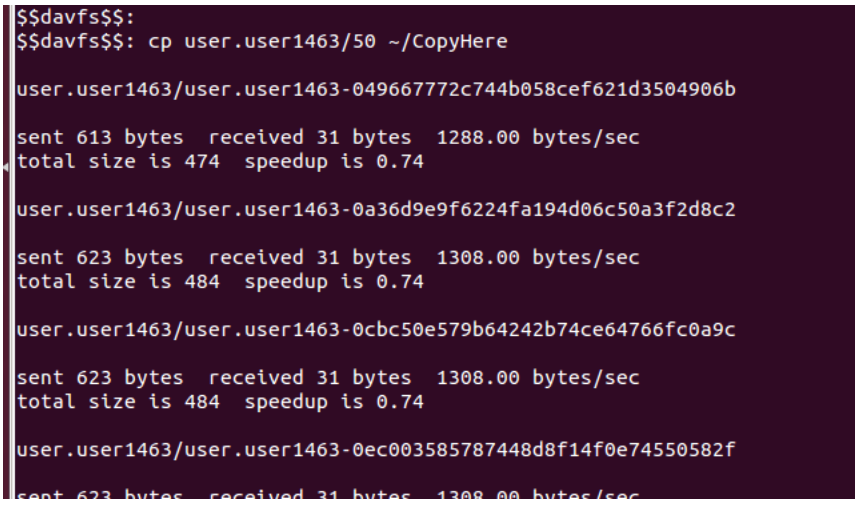

Figure 2: *A typical use of the copy command: copying everything in the directory* "user.user1463" *containing the string* "50" *in its name to the local directory* "~/CopyHere".

Impressions of the Summer School Program

As a student I was very glad to get accepted in this prestigious program. As a master student in theoretical physics I found the opportunity to work on a physics related computing issue challenging, and definitely rewarding. Working on this project has been a great learning experience. I allready have ideas about how I can apply the experience I've gathered during my time at CERN to research some of my other interests, like Monte Carlo simulations. And because of these months I spent here, and the work that I've done, I've taken a deeper interest in computing challenges, like the development of chess engines. I look back at these months as an experience I'm very grateful for having gotten this opportunity and it has given me insights I will be able to use from now on.

Acknowledgements

I would first and foremost like to thank my supervisor, Mario Lassnig, for his support during the last few months. I would also like to thank anyone else who was able to help me along with my project. And I would like to thank CERN for providing me with this wonderful opportunity.