NFS-RODS: A Tool for Accessing iRODS via the NFS Protocol

Presenter: Danilo Oliveira - dmo4@cin.ufpe.br

D.Oliveira, I. Fé, A. Lobo Jr., F. Silva, G. Callou, V. Alves, P. Maciel

Center for Informatics UFPE, Brazil

Stephen Worth
EMC Corporation

Jason Coposky
iRODS Consortium
Agenda

- Introduction
- NFS overview
- NFS-RODS
- Final Remarks
- (Live demonstration)
iRODS is a powerful data grid middleware, plenty of useful features on server side.

However, in order to be useful for final users, it depends on our ability to create client applications that address user’s needs.
Introduction

Many ways for accessing iRODS...

iRODS C-lib

iRODS Explorer  Kanki iRODSClient

Jargon core

Jargon REST  iDROP Desktop  iDROP Desktop
Introduction

How can we access iRODS collections as local folders transparently (hiding the details about iRODS from the clients)?
The NFS-RODS project aims to deliver access to an iRODS environment via typical NFS clients.

This project was created on top of UNFS (user space NFS server), and the iRODS C API

UNFS home page: http://unfs3.sourceforge.net/
Network File System (NFS) is a protocol that defines a distributed file system.

The main goal of the NFS protocol is to turn the remote file access transparent for the O.S. users.
NFS

- Technical details:
  - Transport layer: could be UDP or TCP
  - Session layer: Remote Procedure Call
  - Presentation layer: External Data Representation (XDR) protocol (also developed by Sun Microsystems)
  - Stateless server: the server doesn’t maintain information between successive client requests
NFS and iRODS are similar in some aspects, but, different in others.

The main challenge of this project was how to map the functions specified by the NFS v3 RFC to iRODS calls.
NFSv3 procedures

GETATTR: Get file attributes
SETATTR: Set file attributes
LOOKUP: Lookup filename
ACCESS: Check access permission
READ: Read from file
WRITE: Write to file
CREATE: Create a file
MKDIR: Create a directory
REMOVEDIR: Remove a directory
REMOVE: Remove a file
RENAME: Rename a file or directory
READDIR: Read from directory
LINK: Create link to an object
READDIRPLUS: Extended read from directory
FSINFO: Get static file system information
FSSTAT: Get dynamic file system information
PATHCONF: Retrieve POSIX information
COMMIT: Commit cached data on a server to stable storage
SYMLINK: Create a symbolic link
MKNOD: Create a special device
READLINK: Read from symbolic link
Non-correspondence of attributes

**NFS**

- ftype3 type;
- uint32 nlink;
- size3 size;
- size3 used;
- specdata3 rdev;
- uint64 fsid;
- fileid3 fileid;
- nfstime3 atime;
- nfstime3 mtime;
- nfstime3 ctime;
- mode3 mode;
- uid3 uid;
- gid3 gid;

**iRODS**

```
#define COL_D_DATA_ID 401
#define COL_D_COLL_ID 402
#define COL_DATA_NAME 403
#define COL_DATA_REPL_NUM 404
#define COL_DATA_VERSION 405
#define COL_DATA_TYPE_NAME 406
#define COL_DATA_SIZE 407
#define COL_D_RESC_NAME 409
#define COL_D_DATA_PATH 410
#define COL_D_OWNER_NAME 411
#define COL_D_OWNER_ZONE 412
#define COL_D_REPL_STATUS 413
#define COL_D_DATA_STATUS 414
#define COL_D_DATA_CHECKSUM 415
```

Which are my attributes?
Non-correspondence of attributes

NFS
- atime -> access
- mtime -> modify content
- ctime -> change attributes

iRODS
- COL_D_CREATE_TIME -> creation
- COL_D_MODIFY_TIME -> change attributes

Same value
Non-correspondence of attributes

Our solution:
copy the same value for all fields

NFS
- atime -> access
- mtime -> modify content
- ctime -> change attributes

iRODS
- COL_D_CREATE_TIME -> creation
- COL_D_MODIFY_TIME -> change attributes

Same value
## Change Permissions

<table>
<thead>
<tr>
<th>Most permissive mode unix</th>
<th>iRODS</th>
</tr>
</thead>
<tbody>
<tr>
<td>chmod 7xx</td>
<td>own</td>
</tr>
<tr>
<td>chmod 6xx</td>
<td>own</td>
</tr>
<tr>
<td>chmod 5xx</td>
<td>Read</td>
</tr>
<tr>
<td>chmod 4xx</td>
<td>Read</td>
</tr>
<tr>
<td>chmod 3xx</td>
<td>Write</td>
</tr>
<tr>
<td>chmod 2xx</td>
<td>Write</td>
</tr>
<tr>
<td>chmod 1xx</td>
<td>Read</td>
</tr>
<tr>
<td>chmod 0xx</td>
<td>Null</td>
</tr>
</tbody>
</table>
System Architecture

1 - Client logs into the system using the authentication server

NFS-Client \rightarrow NFS-Calls \rightarrow NFS-RODS

Authentication

Authentication

Authentication

iRODS Calls

iRODS
System Architecture

2 - Client mounts the remote folder and perform NFS calls on the remote folder
System Architecture

3 - NFSRODS server translates each call in iRODS API calls, using the user credentials to authorize access for private folders.
System Architecture

- The authentication server is not mandatory, if we want to access folders inside the `<zone name>/home/public` folder.
Package Diagram
1. Clone the git repository:
   git clone https://github.com/modcs/NFSRODS.git
2. Run “make” on the project folder
3. Install rpcbind
   sudo apt-get install rpcbind
4. Run “./NFSRODS” (runs in background mode)
NFS-RODS: A Tool for Accessing iRODS Repositories via the NFS Protocol

Introduction

iRODS is an open source platform for managing, sharing and integrating data [1]. It has been widely adopted by organizations around the world. iRODS is released and maintained through the iRODS Consortium which involves universities, research agencies, government, and commercial organizations. It aims to drive the continued development of iRODS platform as well as support the fundraising, development, and operation of the iRODS user community.
There is a “.pro” file to open the project in QT-Creator. We are using QT-Creator as IDE, but we don’t use any QT function.
Final remarks

- NFS-RODS allows administrators and users familiar with NFS to interact with iRODS
- iRODS and NFS are not fully compatible, so we had to make some decisions to integrate them in a meaningful way
Live demonstration
Any questions?
Thanks!