Rule-Based Distributed Data Management

Installation

Reagan W. Moore
Arcot Rajasekar
Mike Wan
Wayne Schroeder

{moore,sekaar,mwan,schroede}@diceresearch.org
http://irods.diceresearch.org

UCSD UNC

iRODS Wiki

- http://irods.diceresearch.org
- Descriptions of the technology
- Publications / presentations
- Download
- Performance tests
- Tinderbox system (tracks upgrades)
- irods-chat page

iRODS

Class Exercise

- **http://irods.diceresearch.org**
  - Downloads
    - BSD license
    - Registration / agreement
  - Tar file
    - Installation script (Linux, Solaris, Mac OSX)
    - Automated download of PostgreSQL, ODBC
    - Installation of PostgreSQL, ODBC, iRODS
    - Initiation of iRODS collection

iRODS Installation

Class Exercise

- **Unpack the release tar file**
  - gzip -d irods.tar
  - tar xf irods.tar
- **cd into the top directory and execute**
  - ./irodssetup
- **It will prompt for a few parameters**

irodssetup

- Set up iRODS
- --
- iRODS is a flexible data archive management system that supports many different site configurations. This script will ask you a few questions, then automatically build and configure iRODS.

- There are four main components to iRODS:
- 1. An iRODS server that manages stored data.
- 2. An iCAT catalog that manages metadata about the data.
- 3. A database used by the catalog.
- 4. A set of 'i-commands' for command-line access to your data.

- You can build some, or all of these, in a few standard configurations. For new users, we recommend that you build everything.

iRODS Client Installation

- iRODS configuration setup
- ---
- This script prompts you for key iRODS configuration options.
- Default values (if any) are shown in square brackets [ ] at each
- prompt. Press return to use the default, or enter a new value.

- For flexibility, iRODS has a lot of configuration options. Often
- the standard settings are sufficient, but if you need more control
- enter yes and additional questions will be asked.

- Include additional prompts for advanced settings [no]?

UCSD UNC

iRODS Client Installation

- iRODS configuration (advanced)
- -------------------------------
- iRODS consists of clients (e.g. i-commands) with at least one iRODS
- server. One server must include the iRODS metadata catalog (iCAT).

- For the initial installation, you would normally build the server with
- the iCAT (an iCAT-Enabled Server, IES), along with the i-commands.

- After that, you might want to build another Server to support another
- storage resource on another computer (where you are running this now).
- You would then build the iRODS server non-ICAT, and configure it with
- the IES host name (the servers connect to the IES for ICAT operations).

- If you already have iRODS installed (an IES), you may skip building
- the iRODS server and iCAT, and just build the command-line tools.

- Build an iRODS server [yes]? no

iRODS Client Installation

- iRODS can make use of the Grid Security Infrastructure (GSI)
- authentication system in addition to the iRODS secure
- password system (challenge/response, no plain-text).
- In most cases, the iRODS password system is sufficient but
- if you are using GSI for other applications, you might want
- to include GSI in iRODS. Both the clients and servers need
- to be built with GSI and then users can select it by setting
- irodsAuthScheme=GSI in their .irodsEnv files (or still use
- the iRODS password system if they want).

- Include GSI [no]? no

iRODS Client Installation

- Confirmation
- -------------
- Please confirm your choices.
- --
- GSI not selected

- Build iRODS command-line tools
- --
- Save configuration (irods.config) [yes]?
- Saved.

- Start iRODS build [yes]?

iRODS Client Installation

- Build and configure
- -------------------

- Preparing...

- Configuring iRODS...
-
- Step 1 of 4: Enabling modules...
- properties
-
- Step 2 of 4: Verifying configuration...
- No database configured.
-
- Step 3 of 4: Checking host system...
- Host OS is Mac OS X.
- Perl: /usr/bin/perl
- C compiler: /usr/bin/gcc (gcc)
- Flags: none
- Loader: /usr/bin/gcc
- Flags: none
- Archiver: /usr/bin/ar
- Ranlib: /usr/bin/ranlib
- 64-bit addressing not supported and automatically disabled.

iRODS Client Installation

- Step 4 of 4: Updating configuration files...
- Updating config.mk...
- Created /storage-site/iRODS/config/config.mk
- Updating platform.mk...
- Created /homs/sdc/iRODS/config/platform.mk
- Updating irods.config...
- Updating irodsctl...
- Compiling iRODS...
- Step 1 of 2: Compiling library and i-commands...
- Step 2 of 2: Compiling tests...
- Done!

iRODS Client Installation

- -----

- To use the iRODS command-line tools, update your PATH:
- For csh users:
- set path=(/storage-site/iRODS/clients/icommands/bin $path)
- For sh or bash users:
- PATH=/storage-site/iRODS/clients/icommands/bin:$PATH

- Please see the iRODS documentation for additional notes on how
- to manage the servers and adjust the configuration.

- Change the path name to your installation path

.irodsEnv file

- irodsHost ‘ccirods02.in2p3.fr’
- irodsPort 5580
- irodsHome ‘/workshop/home/user1’
- irodsCwd ‘/workshop/home/user1’
- irodsUserName ’user1'
- irodsZone ’workshop’

User name is “user1”

Password is lyonws09_user1

Full Install

- iRODS configuration:
- ---------------------
- Build an iRODS server? (yes/no) yes
- Include an iCAT catalog? (yes/no) yes

- For security reasons, the build process will create a new iRODS administrator account named 'rods' for managing the system.

- Enter a new password for the iRODS account? (password) xxxxxx

Input Parameters

- **Database configuration:**
- **------------------------**
- **The iCAT uses a database to store metadata. You can build and configure a new Postgres database now or use an existing database.**
- **Build Postgres? (yes/no) yes**
- **You can select the directory for Postgres:**
  - If you are creating a new iRODS installation, select a new directory. Postgres will be automatically downloaded, built, and installed there.
  - If you are upgrading an iRODS installation and wish to re-use an existing database, enter the path to that Postgres directory.
- **Where should Postgres be installed? (directory path) /Astorage-site/Postgres**
- **For security reasons, the new database will create an administrator account for 'reaganmoore' and assign a password.**
- **Enter a password for the new database account? (password) XXXXXXXX**

Check Input Parameters

- The iRODS build and setup is ready to begin.
- iRODS server: build
- account ’demo'
- password ‘demo'
- path '/home/sdsc/iRODS'
- iCAT catalog: build
- Postgres: install a new database
- enable iRODS scripts to start/stop database
- account ’DBadmin'
- password ’UKdemo'
- path '/storage-site/irods/postgresql'
- I-commands: build
- Ready? (yes/no) yes

Installation

Class Exercise

- `Track the completion status of each step:`
- **Preparing...**
- **Installing Postgres database...**
    - Step 1 of 4: Preparing to install...
    - Step 2 of 4: Installing Postgres... About 11 minutes
    - Step 3 of 4: Installing UNIX ODBC... About 26 minutes
    - Step 4 of 4: Setting up Postgres...
    - Step 5 of 4: Setting up iRODS...
- **Configuring iRODS...** About 1 minute
    - Step 1 of 5: Enabling modules...
    - Step 2 of 5: Verifying configuration...
    - Step 3 of 5: Checking host system...
    - Step 4 of 5: Updating configuration files...
    - Step 5 of 5: Cleaning out previously compiled files...
- **Compiling iRODS...** About 3 minutes
    - Step 1 of 3: Compiling library and i-commands...
    - Step 2 of 3: Compiling iRODS server...
    - Step 3 of 3: Compiling tests...

iRODS Source Distribution

- INSTALL.txt
- LICENSE.txt
- Makefile
- README.txt
- Configure
- Vault

- irodsctl
- irodssetup

- COPYRIGHT
- CVS
- bin
- clients
- config
- doc
- install
- installLogs
- lib
- modules
- nt
- scripts
- server

User Configuration

- **To use the iRODS 'i-commands', update your PATH:**
- **For csh users:**
  - set path=(/storage-site/iRODS/clients/icommands/bin $path)
- **For sh or bash users:**
  - PATH=/storage-site/iRODS/clients/icommands/bin:$PATH
- **To start and stop the servers, use 'irodsctl':**
  - irodsctl start
  - irodsctl stop
  - irodsctl restart
- **Add '--help' for a list of commands.**

irodsctl options

- **Usage is:**
  - /storage-site/iRODS/scripts/perl/irodsctl.pl [options] [commands]
- **Help options:**
  - --help   Show this help information
- **Verbosity options:**
  - --quiet   Suppress all messages
  - --verbose   Output all messages (default)
- **iRODS server Commands:**
  - istart   Start the iRODS servers
  - istop   Stop the iRODS servers
  - irestart   Restart the iRODS servers

irodsctl options

- Database commands:

dbstart	Start the database servers
dbstop	Stop the database servers
dbrestart	Restart the database servers
dbdrop	Delete the iRODS tables in the database
dboptimize	Optimize the iRODS tables in the database
dbvacuum	Same as 'optimize'

- General Commands:

start	Start the iRODS and database servers
stop	Stop the iRODS and database servers
restart	Restart the iRODS and database servers
status	Show the status of iRODS and database servers
test	Test the iRODS installation

Environment Variables

- **In home directory**
  - cd ~.irods
  - vi .irodsEnv

Environment File

```
# iRODS personal configuration file.
#
# This file was automatically created during iRODS installation.
#   Created Fri Jan 18 10:01:48 2008
#
# iRODS server host name:
irodsHost ‘ccirods02.in2p3.fr’
# iRODS server port number:
irodsPort 5580
# Home directory in iRODS:
irodsHome '/workshop/home/user1'
# Current directory in iRODS:
irodsCwd '/workshop/home/user1'
# Account name:
irodsUserName ’user1'
# Zone:
irodsZone ’workshop'
```

Directory ~irods/server

Class Exercise

```
•  ls -l
•  total 32
•  drwxr-sr-x    5 asdasd  admin   170 Oct  3 16:10 CVS
•  -rw-r--r--    1 asdasd  admin  8906 Sep 28 16:52 Makefile
•  -rw-r--r--    1 asdasd  admin   281 Sep 12 15:28 README.txt
•  drwxr-sr-x    7 asdasd  admin   238 Oct  3 16:10 api
•  drwxr-sr-x   11 asdasd  admin   374 Oct 15 16:34 bin
•  drwxr-sr-x   12 asdasd  admin   408 Oct 15 16:35 config
•  drwxr-sr-x    7 asdasd  admin   238 Oct  3 16:10 core
•  drwxr-sr-x    7 asdasd  admin   238 Oct  3 16:10 drivers
•  drwxr-sr-x    7 asdasd  admin   238 Oct  3 16:10 icat
•  drwxr-sr-x    5 asdasd  admin   170 Oct 15 16:35 log
•  drwxr-sr-x    7 asdasd  admin   238 Oct  3 16:10 re
•  drwxr-sr-x    4 asdasd  admin   136 Oct  3 16:10 rules
•  drwxr-sr-x    4 asdasd  admin   136 Oct  3 16:10 schema
•  drwxr-sr-x    8 asdasd  admin   272 Oct  3 16:10 test
```

UCSD

UNC

Directory ~irods/server/bin

Class Exercise

```
• $ ls -l server/bin
• total 28176
• drwxr-xr-x   5 reaganmo  admin      170 Jan 18 08:39 CVS
• drwxr-xr-x   5 reaganmo  admin      170 Jan 18 08:39 cmd
• -rwxr-xr-x   1 reaganmo  admin  3604048 Jan 18 10:01 irodsAgent
• -rwxr-xr-x   1 reaganmo  admin  3598516 Jan 18 10:01 irodsReServer
• -rwxr-xr-x   1 reaganmo  admin  3611264 Jan 18 10:01 irodsServer
• -rwxr-xr-x   1 reaganmo  admin  3598024 Jan 18 10:01 irodsXmsgServer
• -rwxr-xr-x   1 reaganmo  admin     1655 Sep 12 15:28 list.pl
• -rwxr-xr-x   1 reaganmo  admin     3400 Sep 12 15:28 vacuumdb.pl
```

Directory ~irods/server/config

Class Exercise

```
• $ ls -l server/config
• total 48
• drwxr-sr-x    5 asdasd  admin  170 Oct  3 16:10 CVS
• -rw-r--r--    1 asdasd  admin  782 Sep 12 15:28 HostAccessControl
• -rw-r--r--    1 asdasd  admin  162 Sep 12 15:28 README.txt
• -rw-r--r--    1 asdasd  admin  665 Sep 12 15:28 irodsHost
• -rw-r--r--    1 asdasd  admin  665 Sep 12 15:28 irodsHost.in
• drwxr-sr-x    3 asdasd  admin  102 Oct  3 16:10 packedRei
• drwxr-sr-x   22 asdasd  admin  748 Oct  3 16:10 reConfigs
• -rw-------    1 asdasd  admin  951 Oct 15 16:35 server.config
• -rw-r--r--    1 asdasd  admin  970 Sep 12 15:28 server.config.in
• -rw-r--r--    1 asdasd  admin    0 Oct 15 16:32 server.config.sav
```

Directory ~irods/server/config/reconfigs

```
• $ ls -l server/config/reconfigs
• total 216
• drwxr-sr-x   5 asdasd  admin    170 Oct  3 16:10 CVS
• -rw-r--r--   1 asdasd  admin   4102 Sep 12 15:28 core.dvm
• -rw-r--r--   1 asdasd  admin    763 Sep 19 15:19 core.fnm
• -rw-r--r--   1 asdasd  admin  14384 Oct  3 12:32 core.irb
• -rwxr-xr-x   1 asdasd  admin    192 Sep 12 15:28 core.irb.1
• -rw-r--r--   1 asdasd  admin    227 Sep 12 15:28 core.irb.2
• -rw-r--r--   1 asdasd  admin    101 Sep 12 15:28 core.irb.3
• -rwxr-xr-x   1 asdasd  admin  14157 Sep 19 15:34 core.irb.orig
• -rw-r--r--   1 asdasd  admin   4102 Oct  3 12:32 core2.dvm
• -rw-r--r--   1 asdasd  admin    763 Oct  3 12:32 core2.fnm
• -rw-r--r--   1 asdasd  admin    690 Sep 12 15:28 core2.irb
• -rw-r--r--   1 asdasd  admin    714 Sep 12 15:28 core3.irb
• -rw-r--r--   1 asdasd  admin    777 Sep 26 10:08 core4.irb
• -rw-r--r--   1 asdasd  admin    269 Sep 12 15:28 misc.irb
• -rw-r--r--   1 asdasd  admin   1275 Sep 12 15:28 nara.irb
• -rw-r--r--   1 asdasd  admin    745 Sep 12 15:28 nvo.irb
• -rw-r--r--   1 asdasd  admin    619 Sep 12 15:28 raja.irb
• -rw-r--r--   1 asdasd  admin    750 Sep 12 15:28 raja2.irb
• -rw-r--r--   1 asdasd  admin   2315 Sep 12 15:28 rajatest.irb
• -rw-r--r--   1 asdasd  admin   1372 Sep 12 15:28 reRules
```

Directory ~irods/clients/icommands/bin

Class Exercise

```
•  $ ls -l clients/icommands/bin
•  total 28296
•  drwxr-sr-x   5 asdasd  admin     170 Oct  3 16:10 CVS
•  -rw-r--r--   1 asdasd  admin      57 Sep 12 15:28 chgCoreToCore1.ir
•  -rw-r--r--   1 asdasd  admin      57 Sep 12 15:28 chgCoreToCore2.ir
•  -rw-r--r--   1 asdasd  admin      52 Sep 12 15:28 chgCoreToOrig.ir
•  -rwxr-xr-x   1 asdasd  admin  436148 Oct 15 16:35 iadmin
•  -rwxr-xr-x   1 asdasd  admin  462880 Oct 15 16:35 icd
•  -rwxr-xr-x   1 asdasd  admin  482544 Oct 15 16:35 ichksum
•  -rwxr-xr-x   1 asdasd  admin  469780 Oct 15 16:35 ichmod
•  -rwxr-xr-x   1 asdasd  admin  488652 Oct 15 16:35 icp
•  -rwxr-xr-x   1 asdasd  admin  385056 Oct 15 16:35 ienv
•  -rwxr-xr-x   1 asdasd  admin     224 Sep 26 11:29 ierror
•  -rwxr-xr-x   1 asdasd  admin  397764 Oct 15 16:35 iexecmd
•  -rwxr-xr-x   1 asdasd  admin  385084 Oct 15 16:35 iexit
```

iCommands

~/irods/clients/icommands/bin

- icd
- ichmod
- icp
- ils
- imkdir
- imv
- ipwd
- irm
- ienv
- ierror
- iget
- iput
- ireg
- irepl
- itrim
- irsync
- ilsresc
- iphymv
- irmtrash
- ichksum
- iinit
- iexit
- iqdel
- iqmod
- iqstat
- iexecmd
- irule
- iuserinfo
- isysmeta
- imeta
- iquest
- imiscsvrinfo
- iadmin

iRODS Components

- Clients
- Persistent state information catalog - iCAT
- Server middleware at each storage system
- Rule engine at each storage system

- Implements server-side workflows composed from micro-services
- Rules control execution of micro-services

iRODS Extensibility

- **Rules**
  - Use default rules for data grid capabilities
  - Administrator modification of pre-packaged rules (turn capabilities on and off)
  - Creation of new rules using existing micro-services
  - Write new micro-services and the rules controlling their execution

iRODS Extensibility

- **State information**
  - Use existing system state information, audit trails
  - Add user-defined metadata (descriptive context)
  - Create schema versions (map persistent state name to a different column in the database)
  - Add new system metadata

iRODS Extensibility

- **Drivers**
  - Add drivers for new storage protocols
  - Mounted Collection interface, add drivers to interact with other data management systems to retrieve information required for operations
- **APIs**
  - Add new client types on top of C-library, Unix i-commands, and Java class library
- **Functionality**
  - Add micro-services
  - Extend Posix I/O by adding functions to framework

Connecting to iRODS Collection

Class Exercise

- **iinit - initiate connection using default parameters specified in the file ~/.irods/.irodsEnv**

```
irodsHost ‘ccirods02.in2p3.fr’
irodsPort 5580
irodsHome ‘/workshop/home/user1’
irodsCwd ‘/workshop/home/user1’
irodsUserName ’user1'
irodsZone ’workshop'
```

- **ienv - lists the contents of the .irodsEnv file**
- **Authentication done using the file ~/.irods/.irodsA**
  - Created when you do an iinit

UCSD UNC

Connect to iRODS

- **$ iinit -h**
- **Creates a file containing your iRODS password in a scrambled form, to be used automatically by the icommands.**
- **Usage: iinit [-ehvVl]**

-e	echo the password as you enter it (normally there is no echo)
-l	list the iRODS environment variables (only)
-v	verbose
-V	Very verbose
-h	this help

Disconnect From iRODS

Class Exercise

- `$ iexit -h`
- Exits iRODS session (cwd) and optionally removes the scrambled password file produced by iinit.
- Usage: iexit [-vh] [full]
- If 'full' is included the scrambled password is also removed.
- -v verbose
- -V very verbose
- -h this help

iRODS File Name Space

Class Exercise

```
$ ils -l
/workshop/home/user1:

$ imkdir nvo
$ imkdir tg
$ imkdir looptest
$ ils -l
```

Do you see the new directories?

iRODS File Name Space

Class Exercise

```
$ ils -l
/workshop/home/user1:
  C- /workshop/home/user1/loopTest
  C- /workshop/home/user1/nvo
  C- /workshop/home/user1/tg
```

```
$ iput ../src/icp.c nvo/icp.c
```

Listing File Information

```
$ ils -l nvo
/workshop/home/user1/nvo:
  user1 0 disk1  3693 2008-01-22.16:59 & icp.c

$ ils -L nvo
/workshop/home/user1/nvo:
  user1 0 disk1  3693 2008-01-22.16:59 & icp.c
/srb/srbcache/test/workshop/home/user1/nvo/icp.c
```

../../../server/bin/stop.pl

../../../server/bin/start.pl

iadmin - Main iRODS Administrator Interface

- **Interactive or command-line interface**
  - A blank execute line invokes the interactive mode, where it prompts and executes commands until 'quit' or 'q' is entered. Single or double quotes can be used to enter items with blanks.
- **Manages**
  - users, user-groups, passwords, resources, resource-groups, directories, database, tokens

iadmin

Class Exercise

• iadmin -h	- Command line
• iadmin	- Interactive mode
• h	- help, list commands
• q	- quit

iadmin - Main subcommands

lu	- list user
lr	- list resource
ls	- list files
lz	- list zone
lg	- list group
lgd	- list group details
lrg	- list resource group
lt	- list token
lf	- list file details
mkuser	- make user
moduser	- modify user
rmuser	- remove user
mkresc	- make resource
modresc	- modify resource
rmresc	- remove resource
mkgroup	- make group
rmgroup	- remove group
atg	- add to group
rfg	- remove from group
atrg	- add (resource) to resource group
rfrg	- remove (resource) from resource group
at	- add token
rt	- remove token
pv	- run a periodic vacuum

iadmin

Class Exercise

• iadmin	- Use interactive mode
• mkuser u2 badtype	- Create new user & user type
• lt	- List tokens for allowed type
• lt user_type	- List allowed user types
• mkuser u2 rodsuser	- Create the user
• lu	- List users
• lu u2	- List user u2
• moduser u2 password [pass]	- set password
• q	- quit

iRODS Storage Name Space

Class Exercise

```
$ ilsresc -l disk1
resource name: disk1
resc id: 10004
zone: workshop
type: unix file system
class: archive
location: ‘ccsrb14.in2p3.fr’
vault: /srb/srbcache/iRODS/workshop
free space:
info:
comment:
create time: 01210760753: 2009-01-27.08:38:23
modify time: 01210761073: 2009-01-27.09:12:32
```

iRODS User Name Space

Class Exercise

```
$ iuserinfo
name: rwmoore
id: 10024
type: rodsadmin
zone: workshop
dn:
info:
comment:
create time: 01210817239: 2009-01-27.10:08:50
modify time: 01210817239: 2009-01-27.10:08:50
member of group: public
member of group: rwmoore
```

Standard Operations

- **The capabilities needed to interact with storage systems**
  - Posix I/O
  - File manipulation
  - Metadata manipulation
  - Bulk operations
  - Parallel I/O
  - Remote procedures
  - Registration

iRODS File Manipulation

```
$ iput -h
Usage : iput [-fkKrvV] [-D dataType] [-N numThreads] [-n replNum]
             [-p physicalPath] [-R resource] [-X restartFile]
               localSrcFile|localSrcDir ...  destDataObj|destColl

Usage : iput [-fkKvV] [-D dataType] [-N numThreads] [-n replNum]
             [-p physicalPath] [-R resource] [-X restartFile] localSrcFile
```

Store a file into iRODS. If the destination data-object or collection are not provided, the current irods directory and the input file name are used. The -X option specifies that the restart option is on and the restartFile input specifies a local file that contains the restart info. If the restartFile does not exist, it will be created and used for recording subsequent restart info. If it exists and is not empty, the restart info contained in this file will be used for restarting the operation. Note that the restart operation only works for uploading directories and the path input must be identical to the one that generated the restart file

iRODS File Manipulation

```
$ iput -h
Options are:
  -f   force - write data-object even if it exists already;
       overwrite it
  -k   checksum - calculate a checksum on the data
  -K   verify checksum - calculate and verify the checksum on the
       data
  -N   numThreads - the number of transfer threads to use. A
       value of 0 means no threading. By default (-N option not
       used) the server decides the number of threads to use.
  -R   resource - specifies the resource to store to. This can be
       specified in your environment or via a rule set up by the
       administrator.
  -r   recursive - store the whole subdirectory
  -v   verbose
  -V   Very verbose
  -X   restartFile - specifies that the restart option is on and
       the restartFile input specifies alocal file that contains
       the restart info.
  -h   this help
```

UCSD UNC

Put a File into the iRODS Collection

Class Exercise

```
$ cd ~/iRODS/clients/icommands/src
$ iput icd.c

$ ils -l
/workshop/home/user1:
   user1        0 disk1           4427 2008-05-14.20:01 & icd.c
  C- /workshop/home/user1/looptest
  C- /workshop/home/user1/nvo
  C- /workshop/home/user1/tg
```

UCSD UNC

Resource Group

Class Exercise

iadmin	- interactive mode
lr	- list resources
lr demoResc	- list demoResc
h mkresc	- list options

```
mkresc disk4 'unix file system' archive
‘ccirods02.in2p3.fr’ /storage-site/Vault2
```

lr	- list resources
atrg dr demoResc	- add resource to group
atrg dr demo2Resc	- add resource to group
lrg dr	- list resource group

iadmin

Class Exercise

- Access irods wiki at

  http://irods.diceresearch.org
- Search for "resource group"
  - Logical aggregation of storage resources
- Read irepl page
  - What happens when files are replicated to a resource group?

iRODS Storage Name Space

Class Exercise

Create Storage Resources

```
$ iadmin mkresc nvoReplResc 'unix file system'
archive ‘ccirods02.in2p3.fr’ /storage-
site/Vaultnvo
$ iadmin mkresc tgReplResc 'unix file system'
archive ‘ccirods02.in2p3.fr’ /storage-
site/Vaulttg
```

Storage Resources

```
List storage resources
$ ilsresc
  disk4
  nvoReplResc
  tgReplResc
  dr (resource group)
```

Integrity Challenges

- **Data grids manage shared collections that are distributed across multiple storage systems and institutions**
  - Data grids are responsible for providing recovery mechanisms for all errors that occur in the distributed environment
  - The number of observed problems is proportional to the size of the collections

Integrity Mechanisms

Class Exercise

- $ irepl -h
- Usage : irepl [-aBMrvV] [-n replNum] [-R destResource] [-S srcResource] [-X restartFile] dataObj|collection ...
- Replicate a file in iRODS to another storage resource.

```
$ irepl -R disk2 foo1
$ ils -l
/workshop/home/user1:
 user1 0 dis1  4585 2007-08-30.14:33 & foo1
 user1 1 disk2 4585 2007-09-18.17:36 & foo1
```

Integrity Mechanisms

- **$ irsync -h**
- **Usage : irsync [-rahsvV] [-R resource] sourceFile|sourceDirectory [....] targetFile|targetDirectory**
- **Synchronize the data between a local copy (local file system) and the copy stored in iRODS or between two iRODS copies. The command can be in one of the three modes:**
  - synchronization of data from the client's local file system to iRODS,
  - from iRODS to the local file system,
  - from one iRODS path to another iRODS path.
- **The mode is determined by the way the sourceFile|sourceDirectory and targetFile|targetDirectory are specified.**
  - Files and directories prepended with 'i:' are iRODS files and collections.
  - Local files and directories are specified without any prependage.

Integrity Mechanisms

- **irsync -r foo1 i:foo2**
  - synchronizes recursively the data from the local directory foo1 to the iRODS collection foo2
- **irsync -r i:foo1 foo2**
  - synchronizes recursively the data from the iRODS collection foo1 to the local directory foo2.
- **irsync -r i:foo1 i:foo2**
  - synchronizes recursively the data from the iRODS collection foo1 to another iRODS collection foo2.
- **Checksums are used to determine whether a file should be synchronized**

Integrity Mechanisms

- **$ ichksum -h**
- **Usage : ichksum [-harvV] [-K|f] [-n replNum] dataObj|collection**
- **Checksum one or more data-object or collection from iRODS space.**
- **Options are:**
  - -f force checksum data-objects even if a checksum already exists
  - -a checksum all replica.
  - -K verify the checksum value in icat. If the checksum value does not exist, compute and register one.
  - -n replNum - the replica to checksum; if not specified checksum all replicas
  - -r recursive - checksum the whole subtree; the collection, all data-objects in the collection, and any subcollections and sub-data-objects in the collection.

UNC

Class Exercise - HELP.looptest

- Make two test collections, and load files from your system

- imkdir loopTest
- imkdir loopTest2
- icd loopTest
- iput ../src/ipwd.c
- iput ../src/iquest.c
- iput ../src/ils.c
- ils -l

iRULE icommand

$ irule -h

```
Usage : irule [--test] [-v] rule inputParam outParamDesc
```

Submit a user defined rule to be executed by an irods server. The first form requires 3 inputs:

1) rule - This the rule to be executed.
2) inputParam - The input parameters for the rule are specified here. If there is no input, a string containing "null” must be specified.
3) outParamDesc - Description for the set of output parameters to be returned. If there is no output, a string containing "null" must be specified.

iRULE Command

Usage : irule [--test] [-v] [-l] -F inputFile [prompt | arg_1 arg_2 ...]

The second form reads the rule and arguments from the file: inputFile

- The first (non-comment) line is the rule. The remaining arguments are interpreted as input arguments for the rule.
- If prompt is the first remaining argument, the user will be prompted for values. The current value will be shown and used if the user just presses return.

- **Otherwise, the arguments are interpreted in two ways**
  - In the first way, the arguments have "label=value" format and only those given label-value pairs are replaced and other pairs are taken from the inputFile. All labels start with *.
  - Alternatively, one can give all arguments as inputs without any labels. In such a case the keyword default can be used to use the inputFile value. Use \ as the first letter in an argument as an escape.

iRULE Command

- The inputFile should contain 3 lines, the first line specifies the rule, the second line the input arguments as label=value pairs separated by % and the third line contains output parameters as labels again separated by %. If % is needed in an input value use %%.
- A value of an input argument can be $. In such a case the user will be prompted. One can provide a default value by giving it right after the $. In such a case, the value will be shown and used if the user presses return without giving a value. The input or the output line can be just be the word null if no input or output is needed.
- An example of the input is given in the file:
-     clients/icommands/test/ruleInp1
- In either form, the 'rule' is either a rule name or a rule definition (which may be a complete rule or a subset).
- To view the output (outParamDesc), use the -v option.
- See ruleInp1 for an example outParamDesc.

iRULE Command

- Options are:
- --test enable test mode so that the micro-services are not executed, instead a loopback is performed
- -F inputFile - read the file for the input
- -l list file if -F option is used
- -v verbose
- -h this help

listColl.ir Rule

```
$cd /storage-site/iRods/clients/icommands/test
$vi listColl.ir
myTestRule | | acGetIcatResults(*Action,*Condition,*B)
   ##forEachExec(*B, msiPrintKeyValPair(stdout,*B)
  ##writeLine(stdout,*K),nop) | nop##nop
*Action=list%*Condition= COLL_NAME =
   '/workshop/home/rods/loopTest'%*K=--------test-test-
   test-------------
*Action%*Condition%ruleExecOut
```

Class Exercise - HELP.looptest

```
/* LISTING AND CHECKSUM */
cd /storage-site/iRODS/clients/icommands/bin
irule -F ../test/listColl.ir
ichksum -r .
irule -F ../test/showicatchksumColl.ir

/* can query iRODS metadata
iquest "select DATA_PATH where DATA_NAME =
        'iquest.c'"
vi
    **/ modify file
irule -F ../test/verifychksumColl.ir
irule -F ../test/forcechksumColl.ir
```

How Many Replicas

- **Three sites minimize risk**
  - Primary site
    - Supports interactive user access to data
  - Secondary site
    - Supports interactive user access when first site is down
    - Provides 2nd media copy, located at a remote site, uses different vendor product, independent administrative procedures
  - Deep archive
    - Provides 3rd media copy, staging environment for data ingestion, no user access

Data Reliability

- **Manage checksums**
  - Verify integrity
  - Rule to verify checksums
- **Synchronize replicas**
  - Verify consistency between metadata and records in vault
  - Rule to verify presence of required metadata
- **Federate data grids**
  - Synchronize metadata catalogs

Resource Group - Load Leveling

$ cd /storage-site/iRODS/server/config/reConfigs

Edit core.irb

To the rule "acSetRescSchemeForCreate" add a random sort after the default resource specification

```
acSetRescSchemeForCreate||msiSetDefaultResc(demoResc,
   null)##msiSetRescSortScheme(random)|nop##nop
```

Make a subdirectory with a few small files

```
mkdir d1
ls > d1/foo1   (etc)
iput -r -R dr d1                    /* are using a resource group
ils -l d1
irm -r d1
```

iCommands

• iinit	initialize access
• imkdir *directory*	make directory
• ils	list files
• ilsresc	list storage resources
• iput *directory file*	put file into iRODS
• iget file	get file from iRODS
• imeta -h	list metadata options

Metadata Manipulation

Class Exercise

- $ imeta -h
- Usage: imeta [-vVh] [command]
- Commands are:
- add -d|C|R|u Name AttName AttValue [AttUnits]   (Add new AVU triplet)
- rm   -d|C|R|u Name AttName AttValue [AttUnits]   (Remove AVU)
- rmw -d|C|R|u Name AttName AttValue [AttUnits]   (Remove AVU, use Wildcards)
- ls   -d|C|R|u Name [AttName]   (List existing AVUs for item Name)
- lsw -d|C|R|u Name [AttName]   (List existing AVUs, use Wildcards)
- qu   -d|C|R|u AttName Op AttVal   (Query objects with matching AVUs)
- cp   -d|C|R|u -d|C|R|u Name1 Name2   (Copy AVUs from item Name1 to Name2)
-
- Metadata attribute-value-units triplets (AVUs) consist of an Attribute-Name, Attribute-Value, and an optional Attribute-Units. They can be added via the 'add' command and then queried to find matching objects.
- For each command, -d, -C, -R or -u is used to specify which type of object to work with: dataobjs (irods files), collections, resources, or users. (Within imeta -c and -r can be used, but -C and -R are the iRODS standard options for collections and resources.)

Metadata Manipulation

Class Exercise

- `$ imeta add  -d foo1 Genealogy Moore`
- `$ imeta add -d foo1 “number of persons” 175,143`
- `$ imeta ls -d foo1`

```
AVUs defined for dataObj foo1:
attribute: Genealogy
value: Moore
units:
----
attribute: number of persons
value: 175143
units:
```

Trash

Class Exercise

- irm - transfers file to the trash
- Trash collection is located at
  - /workshop/trash
- Your directory structure is replicated as files are removed
  - irm foo1
  - /workshop/trash/user1/foo1
- irmtrash removes files from trash

User Level Rules

- **irule -F *rulename*** — **Execute your rule**

- **Rules**

showCore.ir	list current rule base
listColl.ir	list checksums
verifychksumColl.ir	verify checksums
forcechksumColl.ir	update checksums
replColl.ir	replicate collection

Checksum Verification Example

$ more ../test/listColl.ir

First line:

```
myTestRule | | acGetIcatResults(*Action,*Condition,*B)##
forEachExec(*B,msiPrintKeyValPair(stdout,*B) ##
writeLine(stdout,*K),nop) |  nop ## nop
```

Second Line:

```
*Action=list%*Condition= COLL_NAME =
  '/workshop/home/rods/loopTest'%*K=--------FILE-----------
```

Third line:

```
*Action%*Condition%ruleExecOut
```

Core.irb File

Class Exercise

```
$ irule -F showcore.ir
 0    core.acPostProcForPut
     IF ($objPath like /workshop/home/rods/nvo/*) {
       msiSysReplDataObj(nvoReplResc,null)
     }
 1    core.acPostProcForPut
     IF ($objPath like /workshop/home/rods/tg/*) {
    delayExec(<PLUSET>1m</PLUSET>,msiSysReplDataObj(tgReplResc,null),nop)
     }
 2    core.acPostProcForPut
     IF ($objPath like *.mdf) {
       msiLoadMetadataFromFile      [msiRollback]
     }
```

Test Replication

Class Exercise

- **more irodsdemo.txt examples, create another resource**

iadmin mkresc demo3Resc 'unix file system' archive ‘ccirods02.in2p3.fr’ /storage-site/Vault3

../../../server/bin/stop.pl

../../../server/bin/start.pl

Test Replication

Class Exercise

- imkdir nvo
- imkdir tg
- ils -l nvo
- iput -R demoResc ../src/icd.c nvo
- ils -l nvo

- How is this different?

UCSD iRODS SRB UNC

Test Replication
Class Exercise

- ils -l tg
- iput -R demoResc ../src/icd.c tg
- ils -l tg

- How is this different?
- lqstat -l
  - Check that the second copy is made

Standard Micro-services

- Format specific data parsing
- Schema based input
- Schema based output
- Generate a DOI
- Shibboleth & GSI virtual organization
  - Map from virtual organization to the groups/roles within iRODS
- Shibboleth on Ajax rich web client
- High water marks - automated backup
- SRB workspace to compound object for publication in Fedora

iRODS data grid at IN2P3

- irodsHost ‘ccirods02.in2p3.fr’
- irodsPort = 5580
- irodsHome = /workshop/home/user1
- irodsCwd = /workshop/home/user1
- irodsUserName = user1
- irodsZone = workshop

irule Examples

- **Invocation and chaining of remote web services**
- **getObjPositionByName.ir**
  - Accesses a sky catalog, issues a request to convert from object name to object location

```
myTestRule||msiObjByName(*objName,*RA,*DEC,*TYPE)|nop
*objName=$m100
*objName%*RA%*DEC%*TYPE
```

- **getCutOutByPosition.ir**
  - Accesses a sky survey and retrieves an image cutout
- **getCutOutByObjName.ir**
  - Accesses the sky catalog, then gets the image cutout and registers the cutout into an iRODS collection

Delayed Execution

- Delayed rule which is executed every 6 minutes
- starting 1 minute after its submission:

- actestMonPerf||delayExec(<PLUSET>1m</PLUSET><EF>6m</EF>, msiServerMonPerf(default, default),nop)|nop

- msiServerMonPerf is executed every 6 minutes.

- EA - execAddress - host where the delayed execution needs to be performed
- ET - execTime - absolute time when it needs to be performed.
- PLUSET - relExeTime - relative to current time when it needs to execute
- EF - execFreq - frequency (in time widths) it needs to be performed.

Delayed Execution

The format for EF is quite rich:
 * The EF value is of the format:
 *   nnnnU <directive> where
 *   nnnn is a number, and
 *   U is the unit of the number (s-sec,m-min,h-hour,d-day,y-year),
 * The <directive> can be for the form:
 *   <empty-directive>   - equal to REPEAT FOR EVER
 *   REPEAT FOR EVER
 *   REPEAT UNTIL SUCCESS
 *   REPEAT nnnn TIMES   - where nnnn is an integer
 *   REPEAT UNTIL <time> - where <time> is of the time format supported by checkDateFormat function
 *   REPEAT UNTIL SUCCESS OR UNTIL <time>
 *   REPEAT UNTIL SUCCESS OR nnnn TIMES
 *   DOUBLE FOR EVER
 *   DOUBLE UNTIL SUCCESS   - delay is doubled every time.
 *   DOUBLE nnnn TIMES
 *   DOUBLE UNTIL <time>
 *   DOUBLE UNTIL SUCCESS OR UNTIL <time>
 *   DOUBLE UNTIL SUCCESS OR nnnn TIMES
 *   DOUBLE UNTIL SUCCESS UPTO <time>

UCSD

UNC

irule Examples

- **irodsdemo.txt**
  - Lists examples for delayed execution
- **HELP.looptest**
  - Lists examples for checksums, copying, replicating, sending e-mail, purging files
  - Lists the irule tests that are validated for loops, remote execution, metadata extraction, web services