OSG STORAGE/IRODS INTEGRATION

Tanya Levshina Fermilab
Ashu Guru

Open Science Grid

Why am I giving this talk?

- To provide a status report?
  - Not really, though it is nice to explain what we are working on.
- To describe a new technology?
  - Probably not, iRODS is well known and has been around for some time.
- To identify a new problem?
  - Not likely, many of the OSG opportunistic users are aware of the difficulties managing space and handling sizeable data movement
- To recruit guinea pigs?
  - YES! We are very interested in people who want to try our new approach.
- To solicit feedback and identify new use cases?
  - YES!

Outline

- Use Cases
- Motivation and Problem Statement
- Requirements
- Proposed Solution
- iRODS overview
- High Level Architecture
- Proof of concept
- Conclusion
- Future Work

Use Cases

- SCEC, Fly's Eye experiment and others: Need to find sites for pre-staging 1-5 TB of data per site
- NEES – Needs to handle output: 1 GB of output file per job, total output: ~ 4 TB; Total number of jobs: 3,780. Job duration: 6-12 hours
- SLAC Theory Group (Phenomenology) – Needs to handle output: 2 – 3GB output file per job; 500 jobs submitted at a time, 8-12 hours per job
- Common questions:
  - How to find sites that will host sizeable amount of data?
  - Is there a tool that allows to replicate data/migrate pre-staged data?
  - What should one do with the output files without bringing down a submission node and without causing 12 hours job to fail only because space/storage is not available?
  - How to keep track of all the output files on OSG SEs?

Motivation and Problem Statement

One of the goals of the OSG is to provide the Virtual Organizations (VOs) with opportunistic usage of grid resources. Storage is one of the essential resources. The OSG initiated production scale *Opportunistic Storage* provisioning and usage on all OSG sites.

The major complaints of the VOs relying on public storage are the following:

- most of the sites do not support dynamic storage allocation and do not have tools for automatic management;
- the VOs that rely on opportunistic storage have difficulties finding an appropriate storage, verifying its availability and monitoring its utilization;
- the involvement of a Production Manager, Site Admins and VO support personnel is required to allocate or rescind storage space.

OSG ET Requirements

- Allow the OSG Production manager to manage public storage allocation across all the participating sites.
- Impose minimal burden on the participating sites.
- Simplify SE selection for data storage.

Proposed Solution

- Use iRODS as a resource management and data movement service for the OSG public storage.
- Integrate iRODS with the OSG Storage Elements.
- Run iRODS as SaaS. Deploy it on a central node along with iCAT catalog.

Why iRODS?

- The Integrated Rule-Oriented Data System (iRODS) is developed by the Data Intensive Cyber Environments research group and collaborators.
- iRODS implements a policy-based data management framework.
  - handles various objects (resources, collections and files)
  - each object has a set of properties (metadata) associated with it
  - properties are enforced by polices (set of Rules)
  - rules trigger a chain of actions (micro-services). A chain of actions may include recovery from failures and notification.
  - Provides means to set quota limit and enforce quota management
- iRODS performs transfers by
  - using implementation specific protocol to access POSIX compliant resources
  - using an external driver to Mass Storage. The driver should implement "put" and "get" methods to transfer entire files. File transfer is performed in two steps (disk cache is needed)
- The Metadata Catalog (iCAT) stores complete state information about the system in a database. iCAT contains information about resources, resource usage, quotas and users. It also serves as metadata catalog for users data collections.
- Widely used by scientific community (Biology, Environment , Physical Sciences, Geosciences, etc)

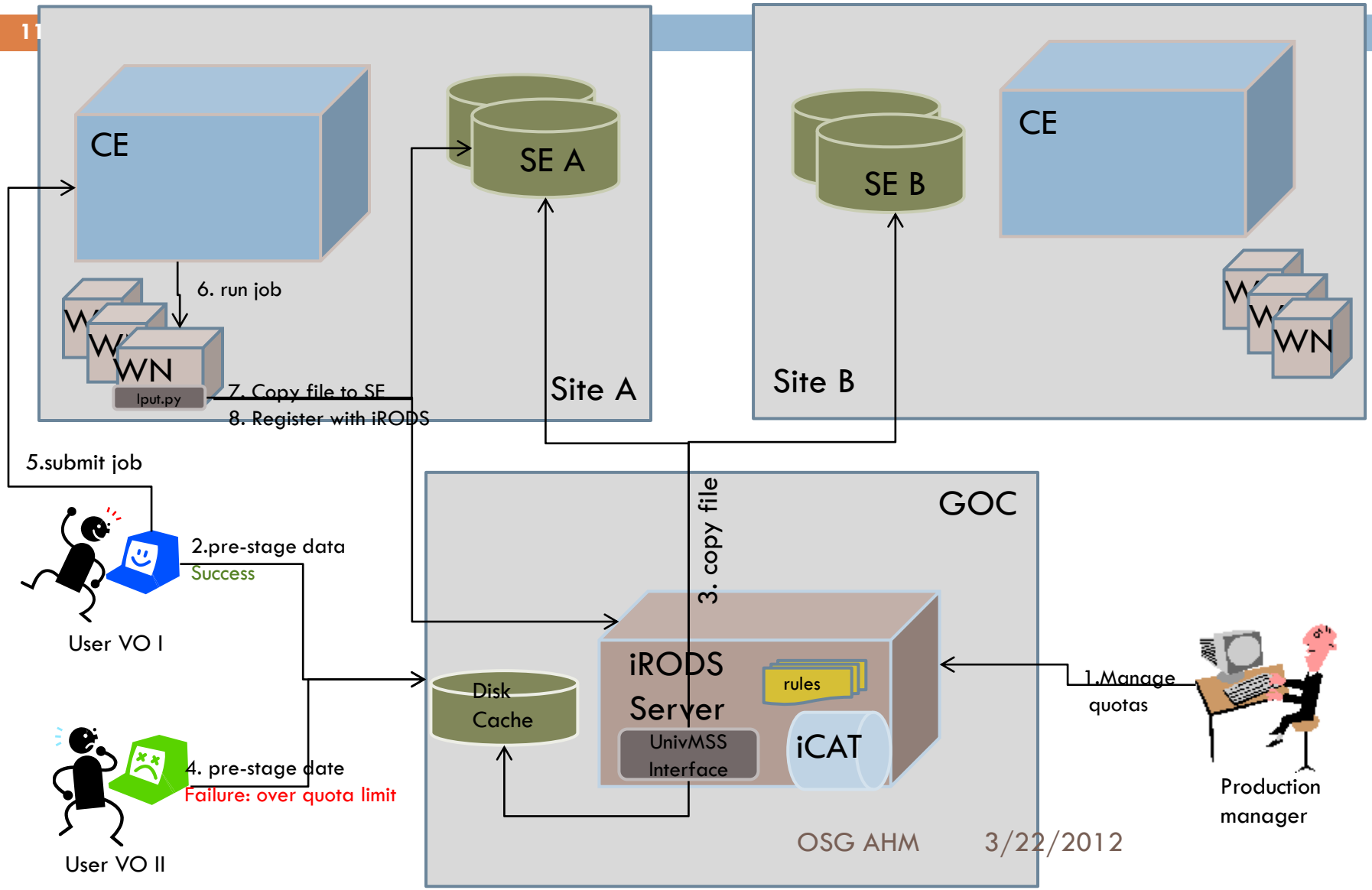
Phase I: Proof of Concept (I)

- Build, install and configure GSI-enabled iRODS
- Integrate iRODS with OSG SEs by using the existing srm-client
- Acquire/implement iRODS rules and micro-services that enforce the following quota management capabilities and example policies:
  - Set quota per resource and group (VO)
  - Prevent users from uploading files if quota limit is reached
  - Delete files from a SE in order to comply with changed quota limit
  - Send notification about success/failure of the actions
- Register Engage VO as Engage group.
- Register several users (user name, email address, DN).
- Assign these users to Engage group of iRODS.
- Register a couple of sites (UCSDT2 & Nebraska)

Phase I: Proof of Concept (II)

- Perform functionality tests:
  - iRODS can authenticate user with x509 certificates
  - A Production Manager can set/modify quota per resource and group
  - A user can upload/download/delete files to/from the OSG SE via iRODS
  - A user can not write via iRODS if quota limit on resource is reached
  - A user can register a file with iRODS after it was successfully copied to a SE. That is typical use case for the interaction of a job with iRODS from a worker node (WN).
  - iRODS can delete files from storage if needed
  - iRODS can send notification

High Level Architecture



Setting Group/Resource Quota

- For each resource/VO group a Production Manager sets a quota. The information about total available public space on a specific site is provided by a Site Administrator.
- A Production Manager decides how much space needs to be allocated for a particular group (VO) on each resource. The space allocation could be changed at anytime.
- A Production Manager sets quota using:

  ***iadmin sgq* Group ResourceName Value**
- The rule that handles enforcement of quota is enabled in iRODS core rules.
- The quota limit change triggers the execution of the rule:
  - Checks if quota is exceeded per group/resource
  - If so, deletes files until space utilization is under the limit
  - Sends email notifications to the owners of deleted files
  - Sends report to irods admin
- A production manager monitors the current space utilization using ***iquota***

Conclusion

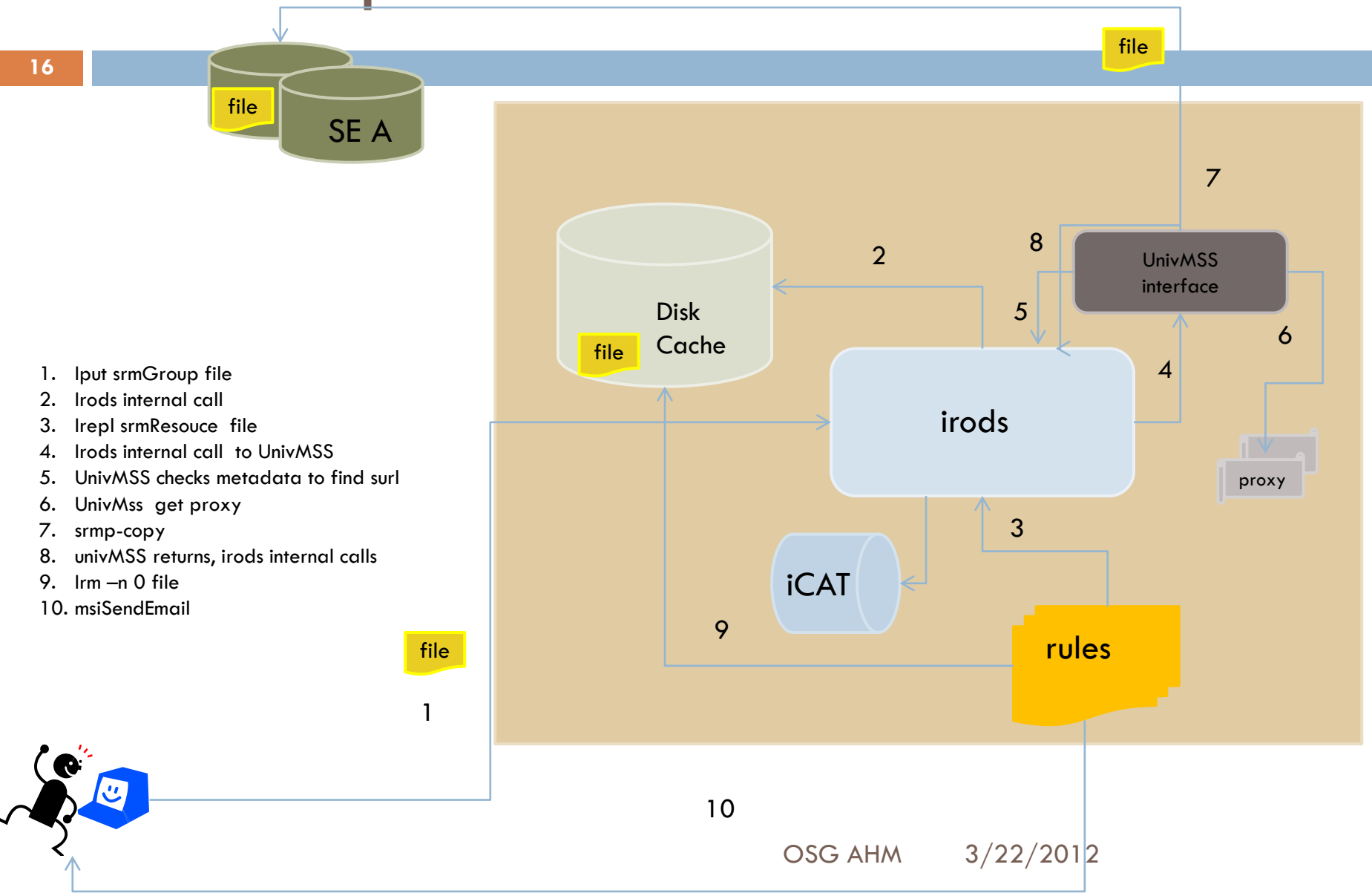
- With our prototypical deployment, we have demonstrated the feasibility of managing public storage at the OSG sites with iRODS.
  - A Production Manager can manage resource allocations at remote sites between various VOs.
  - No actions are required from the sites after initial allocation of resources.
  - A user can upload and download files from a user laptop or a worker node using iRODS commands and in-house developed scripts.

Future Work

- Before we can proceed to the next step we need to test performance and scalability of the current installation.
- We have identified the following goals for phase II:
  - Test basic failure condition and recovery.
  - Test resource allocation and management with two VOs and several sites. Modify rules if needed.
  - Identify a VO, users that can benefit from access to public storage via iRODS.
  - Negotiate with the OSG sites.
  - Help a selected user to adopt a new workflow.

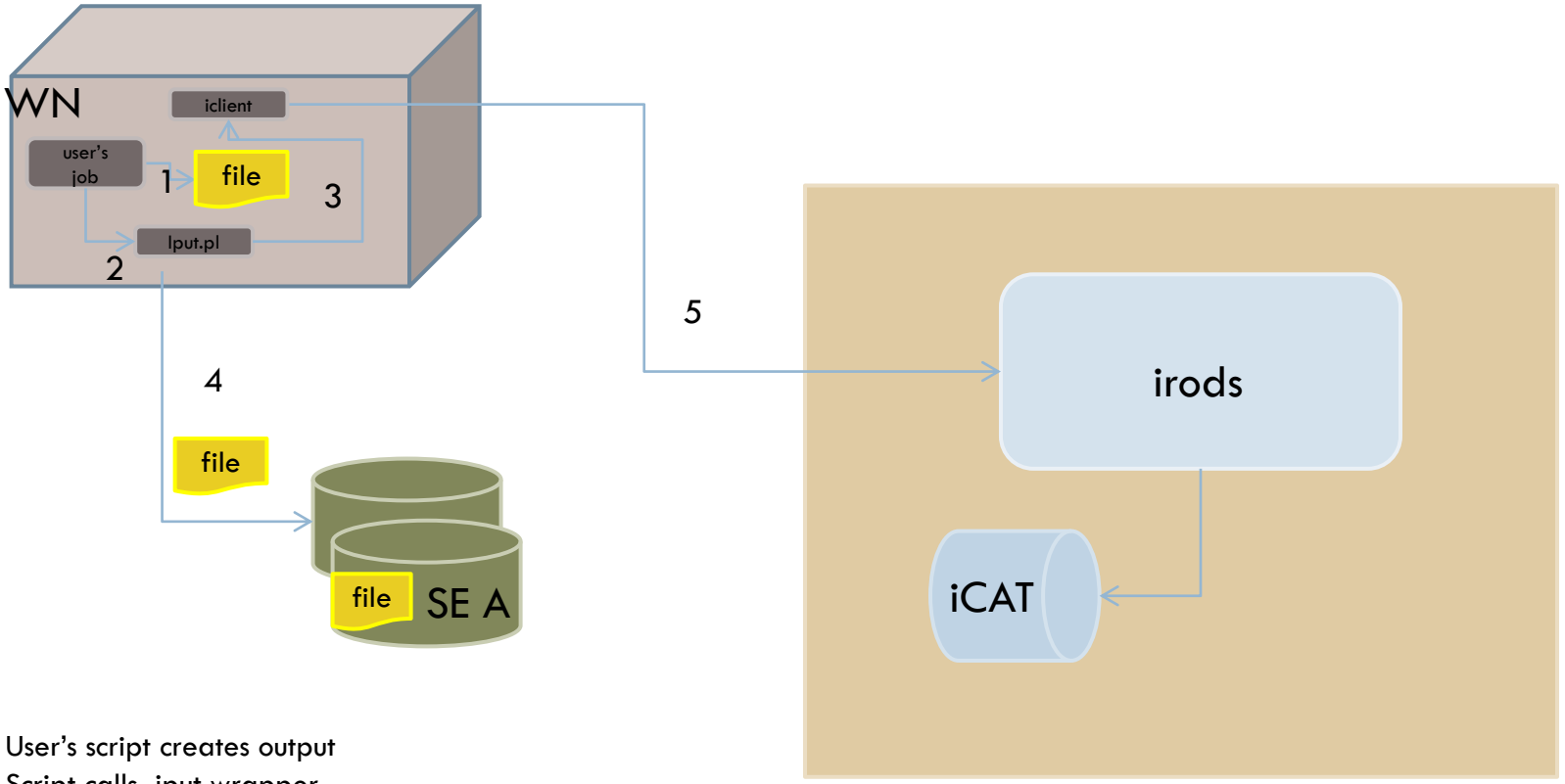
Additional Slides

Data Upload Workflow



1. Iput srmGroup file
2. Irods internal call
3. Irepl srmResouce file
4. Irods internal call to UnivMSS
5. UnivMSS checks metadata to find surl
6. UnivMss get proxy
7. srmp-copy
8. univMSS returns, irods internal calls
9. Irm –n 0 file
10. msiSendEmail

Data Upload From A Worker Node



1. User's script creates output
2. Script calls iput wrapper
3. Wrapper issues calls to irods to find out if local SE exists and has enough space
4. Wrapper copies file to SE
5. Wrapper registers file wit irods (ireg)

iRODS Service Certificate

- Obtain iRODS service certificate
- Register iRODS service certificate as a member with participating VOs. Currently (HCC, Fermilab and Engage).
- Create and periodically update proxy certificate for all the VOs using cronjob or other means.
- Proxy files are named <voname>_proxy and located in ~irods/.globus directory.

Setting IRODS Resources (create_srm_resource.sh)

- Creates a resource group, e.g *osgSrmGroup*
- Pulls information from bdii for a particular VO. Creates compound resources of type "MSS Universal Driver" for all participating sites with Storage Elements. Resource name is set to Site Names: *Firefly, UCSDT2, AGLT2*
- Adds metadata from bdii for each resource. Metadata contains – surl, end path for each VO and local path if exists, eg:

imeta ls -R UCSDT2

attribute: hcc
value: /hadoop/hcc/irods/,/hadoop/hcc/irods
----
attribute: Engage
value: /hadoop/engage/irods/,/hadoop/engage/irods
----
attribute: surl
value: srm://bsrm-1.t2.ucsd.edu:8443/srm/v2/server

- Sets quota to "1 byte" for all resources
- Creates cache resource of type "unix file system" , e.g *diskCache*
- Adds all these resources to the resource group

List of resources

ilsresc

- UC_ITB
- BNL-ATLAS
- NYSGRID_CORNELL_NYS1
- Vanderbilt-ITB
- MWT2
- TTU-ANTAEUS
- FNAL_FERMIGRID_ITB
- diskCache
- Purdue-RCAC
- UCSDT2
- SPRACE
- Vanderbilt
- FNAL_FERMIGRID
- Firefly
- FNAL_IRODS_TEST1
- UConn-OSG
- GLOW
- CIT_CMS_T2
- Nebraska
- UCR-HEP
- WT2
- FNAL_GPGRID_1
- osgSrmGroup (resource group)

Setting iRODS Users and Groups

- Create an iRODS group for each participating VO. A group name should be the same as a VO name.
- User information can be populated by contacting corresponding VOMS instance and extracting user DN, email address (create_user.sh)
  - It is unclear how to create a user name in iRODS when we start automatic registration of users. One way of doing it is to add an attribute to VOMS (irods login). The similar approach is used by some major VOs for afs login.

iuserinfo tlevshin

name: tlevshin

id: 10519

type: rodsuser

zone: osg

info: tlevshin@fnal.gov

comment:

create time: 01329602644: 2012-02-18.16:04:04

modify time: 01329602680: 2012-02-18.16:04:40

GSI DN or Kerberos Principal Name: /DC=org/DC=doegrids/OU=People/CN=Tanya Levshina 508821

tlevshin member of group: Engage

Modification of univMSSinterface.sh

This script is executed by iRODS when ***irepl, irm*** and ***ireg*** commands are issued. It performs the following actions:

- Determines user's group and finds appropriate proxy service certificate using irods client commands
- Gets surl and end path from resource metadata
- So far we have implements the following methods:
  - syncToArch (srm-copy local_cache surl)
  - stageToCache (srm-copy surl local_cache)
  - rm (srm-rm surl)
  - mkdir (srm-mkdir surl)
  - stat (srm-ls surl)

Data Management Rules

- Quota_Management rule:
  - Checks if quota is exceeded per group/resource
  - If so, deletes files until space utilization is under the limit
  - Sends email notifications to the owners of deleted files
  - Sends report to irods admin
- Replication Rule:
  - Finds all the files that are located on a disk cache but not in any storage. Selects best storage for the file, replicates the file, deletes it from disk cache.
  - Sends email notification that file is available on a specific resource
- Disk Cache Clean up rule:
  - Periodically checks disk cache and deletes files that have been replicated. This situation occurs when user upload file to a particular compound resource using "***iput compoundResc file***"

File Registration From a Worker Node

Use Case: A user wants to submit a job to the grid and needs upload data to the local SE from a worker node

- We will need iRODDS client command to be installed on worker nodes (for now it is shipped as a tar.gz file with a job).
- A wrapper script has to be shipped with a job. This script allows to:
  - check if local storage exists (resource metadata info)
  - check if file could be stored locally (quota limit)
  - check if local mount is available or get surl (resource metadata info)
  - upload file using appropriate copy command
  - register this file with iRODS using ***ireq*** command

Example of the Job Submission File

- Job submission file:

```
#test job
executable = irods_iput_test.sh
#irods user enviroment, contains information about irods server , host, username , home are  etc
environment = "irodsEnvFile='irodsEnv' Process=$(Process)  Cluster=$(Cluster)"
# irods client command, wrapper script
transfer_input_files = irods_client.tar.gz,client.tar,irodsEnv
requirements    = (arch == "X86_64")
log = irods_test_run_$(Cluster)_$(Process).log
output = irods_test_run_$(Cluster)_$(Process).out
error = irods_test_run_$(Cluster)_$(Process).err
x509userproxy = /tmp/x509up_u4461
notification = Never
should_transfer_files = YES
when_to_transfer_output = ON_EXIT
arguments =
queue
```

Test Job Example

- Test job script

```
#!/bin/bash
#untar irods related  execs
tar xfz irods_client.tar.gz
tar xvf client.tar
#create test file
fn=irods_test_${GLIDEIN_ResourceName}
dd if=/dev/urandom bs=1024 count=1024 of=${fn}.${Cluster}_${Process}
echo "Running at the site ${GLIDEIN_ResourceName}" >>irods_commands.${Cluster}.${Process}.log 2>&1
export PATH=bin/:$PATH
client/iput.py -d  ${fn}.${Cluster}_${Process} >>irods_commands.${Cluster}.${Process}.log 2>&1
rm ${fn}.${Cluster}_${Process}
exit $?
```

File Listing Example

ils -l

/osg/home/tlevshin:

tlevshin 0 Nebraska 1048576 2012-02-22.21:46 & irods_test_AGLT2.2858468_0
tlevshin 0 Nebraska 2012 2012-02-22.14:15 & irods_test_AGLT2.2858469_0
tlevshin 0 Nebraska 2012 2012-02-22.14:29 & irods_test_AGLT2.2858470_0
tlevshin 0 Nebraska 1048576 2012-02-22.22:31 & irods_test_AGLT2.2858473_0
tlevshin 0 UCSDT2 1048576 2012-02-28.14:54 & irods_test_AGLT2.2960375_0
tlevshin 0 UCSDT2 1048576 2012-02-28.15:02 & irods_test_AGLT2.2966498_0
tlevshin 0 UCSDT2 1048576 2012-02-28.15:13 & irods_test_AGLT2.2971401_0
tlevshin 0 Nebraska 1048576 2012-02-22.21:44 & irods_test_FNAL_GPGRID_1.2858465_0
tlevshin 0 Nebraska 2012 2012-02-22.14:11 & irods_test_MWT2_UC.2858467_0
tlevshin 1 Nebraska 735804 2012-02-22.21:43 & test_101
tlevshin 1 Nebraska 1048576 2012-02-23.13:33 & testfile_UCSDT2_1
tlevshin 1 Nebraska 20971520 2012-02-23.13:59 & testfile_UCSDT2_2
tlevshin 1 UCSDT2 1048576 2012-02-27.17:19 & testfile_UCSDT2_4
tlevshin 1 UCSDT2 20971520 2012-02-27.17:20 & testfile_UCSDT2_5

File Listing with srm-ls

You can see file via srm-ls as well:

srm-ls srm://bsrm-1.t2.ucsd.edu:8443/srm/v2/server?SFN=/hadoop/engage/irods/home/tlevshin/irods_test_AGLT2.2971401_0

srm-ls 2.2.2.2.0 Wed Dec 14 11:45:28 PST 2011

SRM-CLIENT*REQUEST_STATUS=SRM_SUCCESS

SRM-CLIENT*SURL=/hadoop/engage/irods/home/tlevshin/irods_test_AGLT2.2971401_0

SRM-CLIENT*BYTES=1048576

SRM-CLIENT*FILETYPE=FILE

SRM-CLIENT*FILE_STATUS=SRM_SUCCESS

SRM-CLIENT*FILE_EXPLANATION=Read from disk..

SRM-CLIENT*FILELOCALITY=ONLINE