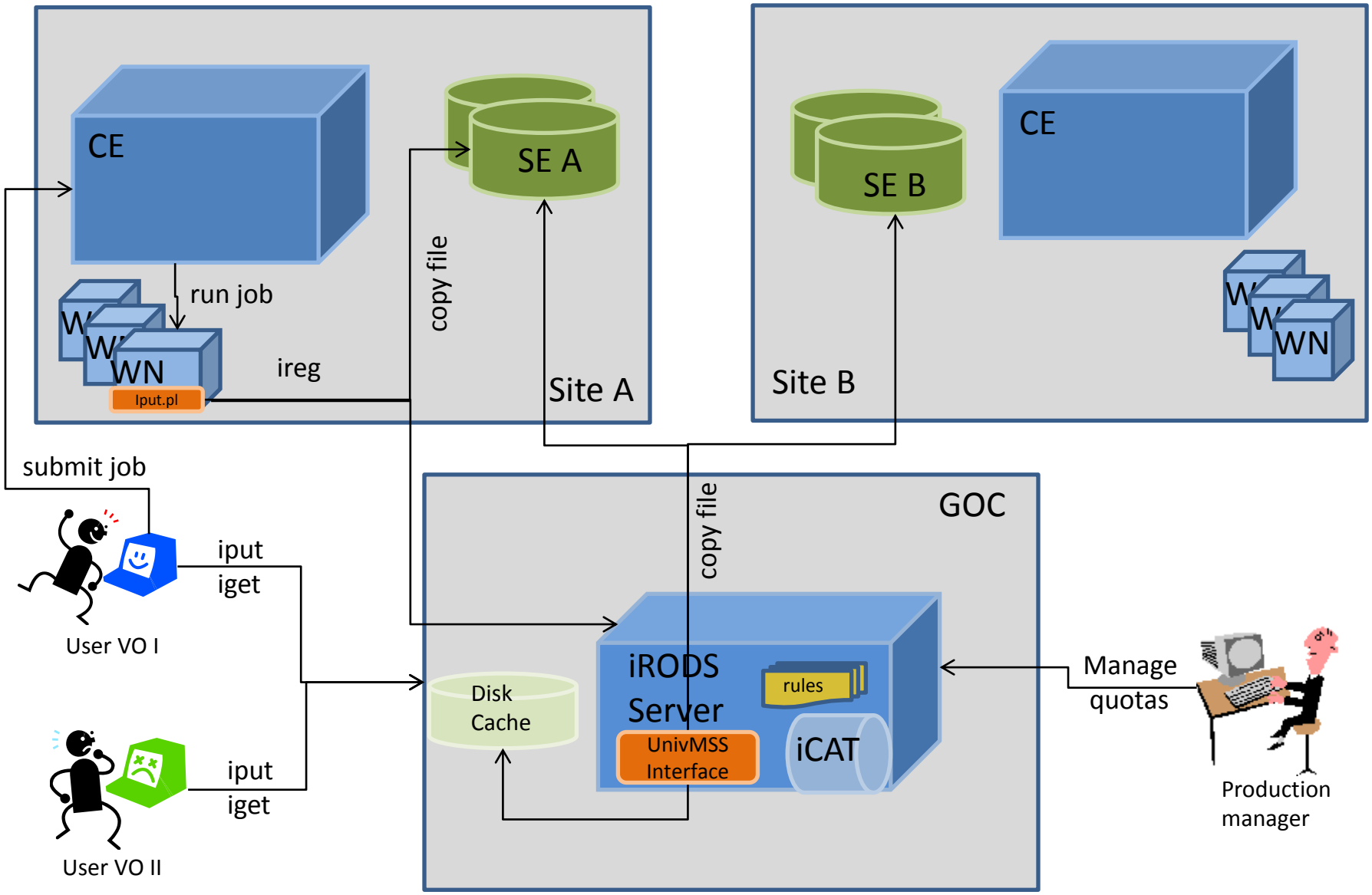
OSG Storage/iRODS Integration

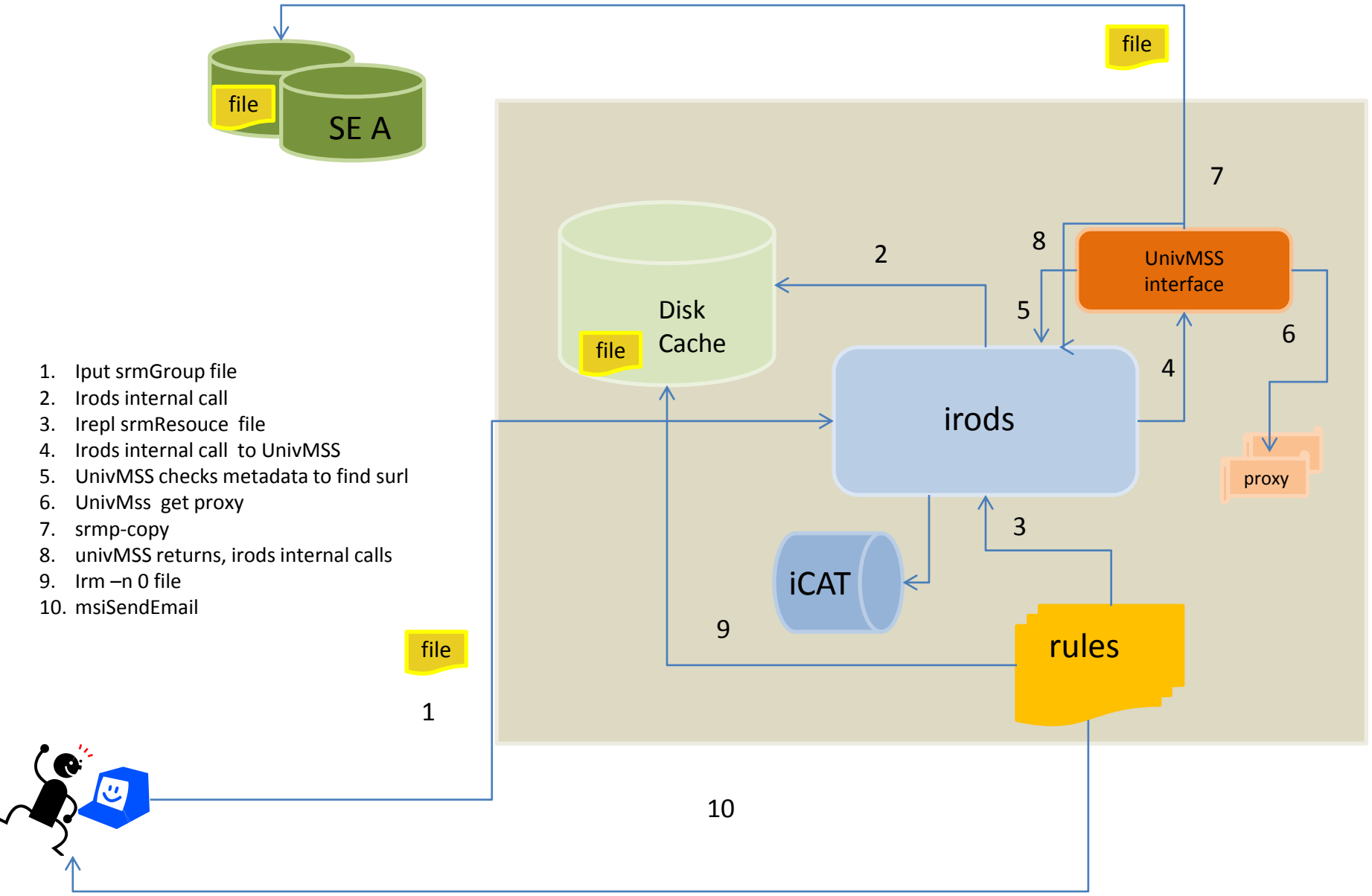
Tanya Levshina

Ashu Guru

High Level Architecture

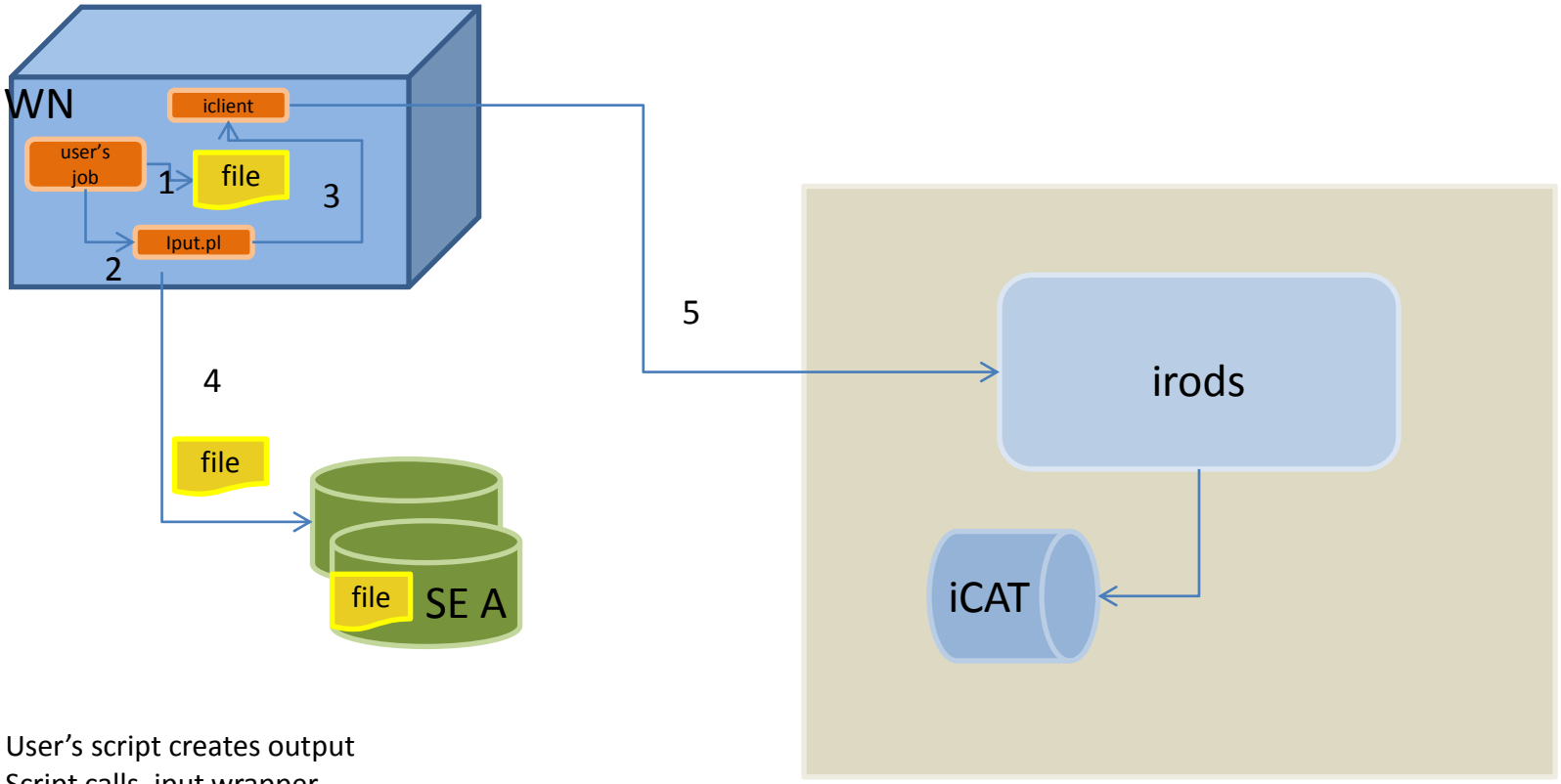


Data Upload Workflow



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

Setting Group/Resource Quota

- For each compound 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
- The rule that handles enforcement of quota (***acRescQuotaPolicy***) should be enabled in iRODS core rules.
- iRODS doesn't recalculate the current quotas, so the periodic rule that triggers current usage and quota calculation should be registered with iRODS as deferred rule. It is using ***msiQuota*** microservice.

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 irods client command to be installed on worker nodes (for now it is shipped as a tar.gz file with a job). We will have to rebuild irods with static libs because there are some site where a particular globus library is missing.
- In the future we can use condor plugin mechanism to manage data movement from a worker node, for the time being a wrapper script is 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 and a test job (engage submit node)

- 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 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
C- /osg/home/tlevshin/Engage
C- /osg/home/tlevshin/test
C- /osg/home/tlevshin/test_106

You can see file via srm 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*

Bugs, questions, requirements (build & installation)

- The only way to build iRODS with globus is to install and build globus toolkit from scratch on the target machine. The attempt to install globus from rpm and build iRODS has failed.
- The setting of GSI environment variables is missing, the following needs to be added to scripts/perl/irodsctl.pl:

```
$ENV{'X509_USER_CERT'} = '/etc/grid-security/irodscert.pem';
$ENV{'X509_USER_KEY'} = '/etc/grid-security/irodskey.pem';
```

- Patch has been provided by developers to build code on 32bit node
- In OSG software is distributed as rpm. We will need at least iRODS client command to be packaged as rpm, so it could be included in OSG Client and Worker Node Client distributions.

Bugs, questions, requirements (rules & microservices)

- So far we were unable to find any rules or microservices that allow selection of the “best” resource automatically.
- With new version (3.0) it is unclear how to handle exception within rule workflow.
- For some reason the rules that are executed successfully when invoked manually are failing when an attempt to upload them as deferred execution rules was made.
- We can not find any description how disk cache is managed in general case, for now we have rules that delete any files that are already replicated.

Bugs, questions, requirements (srm driver & certificate handling)

- It will be very beneficial if srm driver is implemented within iRODS. For the time being we are using just srm bestman client commands under univMSSInterface.sh to manage data movement.
- We are not sure that current procedure of handling certificates is optimal (certificate delegation could be a better approach).
- An OSG users are members of multiple VOs subgroups. iRODS doesn't allow to have group hierarchy that could be helpful for this case.