OSG STORAGE/IRODS INTEGRATION

Open Science Grid

Tanya Levshina

Ashu Guru

Yaling Zheng

Fermilab

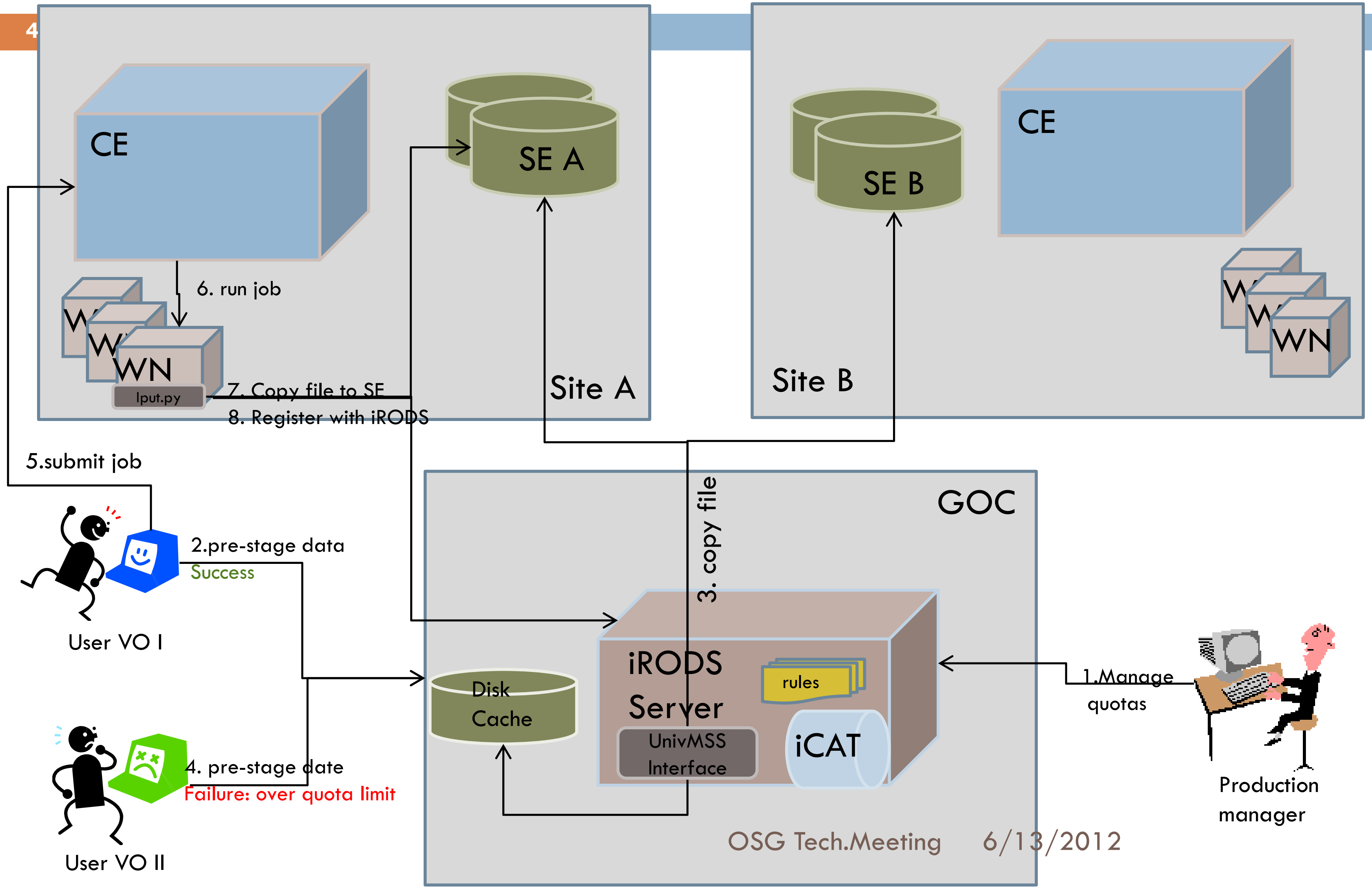
OSG ET Requirements

- 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.

iRODS Deployment

- iRODS server
  - Version 3.0 GSI enabled
  - Hardware:
    - Purchased/installed in Jan 09 – out of warranty
    - Intel® Xeon™ CPU 3.60GHz
    - RAM 4GB
    - 8 cores
    - 1TB disk
    - 64bit
  - KVM VM:
    - 2 QEMU Virtual CPUs (0.9.1)
    - 229 GB disk
    - 2GB memory
    - 32bit
  - iRODS clients 3.0 are installed on multiple nodes at Fermi, Nebraska, Renci

High Level Architecture



Phase I: Proof of Concept

- Phase I is complete:
  - Integrated iRODS with OSG SEs by using the existing srm-client (univMSSInterface.sh)
  - Registered SEs with the following metadata:
    - Status (up/down)
    - Supported group (VO)
    - Quota per group
    - SURL
    - Path/local path
  - Implemented iRODS rules that enforce the following quota management capabilities and policies:
    - 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
  - Implemented selection of appropriate SE when data are uploaded or replicated to a specific group:
    - Upload: Find SE that supports a VO, is up and has sufficient space, upload the file and delete it from disk cache
    - Replicate: Select all the SEs that supports a VO, are up, do not already have a file requested for replication and have sufficient space, replicate the fille and delete it from disk cache
  - Implemented data workflow from the worker node using condor plugin mechanism
  - Implemented irods client/wrapper script installation on the worker node using glidein frontend configuration
  - Registered two VOs (Engage and HCC)
  - Registered multiple users (user name, email address, DN). Users are assigned to a group (VO).
  - Implemented a periodic rule that checks status of a SE and marks its status as up/down depending on the result

iRODS Users, Groups, Resources and Quota Management

- Preparation steps to use the OSG Public Storage
  - A VO Admin contacts a Production Manager and ask for access to OSG Public Storage.
  - A Production Manger asks a VO Admin to register iRODS service certificate in VOMS.
  - A Production Manager decides how much space should be allocated for this VO at the sites that supports it
- Automatic/manual registration of VO users and SEs
  - A Production Manager may execute a script that will register all current VO members with iRODS or register subset of users manually
  - A Production Manager may executes a script that will register all SEs that supports this VO or register susbset of SEs manually
  - A Production Manager sets quota for each resource and this VO group. The information about total available public space on a specific site is provided by a Site Administrator. The space allocation could be changed at anytime.
- 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
- A Production Manager monitors the current space utilization

User Level Data Management (I)

- pre-stage file to a specific SE:

  ```
  iput –R Nebraska my_file
  ```

- pre-stage file to some SE:

  ```
  iput –R osgSrmGroup my_file
  ```

- download file from SE:

  ```
  iget my_file
  ```

- delete file from SE:

  ```
  irm –f my_file
  ```

- replicate file from one SE to all other available SEs:

  ```
  irule –F  $IRODS_LOCATION/client/…/safereplicate2.r /coll/file
  ```

- list file detailed information :

  ```
  ils –l my_file
  ```

User Level Data Management (II)

- login on submission node
- create job description file:

```
transfer_input_files = irodse://<user_name>@gw014k1.fnal.gov:1247?/<collection>/<file_name>
output_destination = irodse://<user_name>@gw014k1.fnal.gov:1247?/<collection>/<file_name>
+UsesiRODS=True
Requirements = (GLIDECLIENT_Group =?= "irodsft" && TARGET.irodsversion=?=1)
```

- submit job
- the job starts on a worker node on a site where a pilot is running and irods software is already installed. It will:
  - check via iRODS the location of the input file
  - download file using srm client from the SE
  - check via iRODS where to upload output file (finds the ‘best resource’ (closest first then space available)
  - upload file to SE using srm client command
  - register file with iRODS
- One can specify a particular SE for output file, eg:

```
transfer_input_files = irodse://<user_name>@gw014k1.fnal.gov:1247?Nebraska/<collection>/<file_name>
```

Frontend Service Configuration Modification

- Specify “irodsoft” group
  - define attributes:

```
<attrs>
<attr name="UsesiRODS" glidein_publish="True" job_publish="True" parameter="True" type="string" value="True"/>
<attr name="irodsversion" glidein_publish="True" job_publish="True" parameter="True" type="int" value="3"/>
</attrs>
```

  - define additional files files:

```
<files>
    <file absfname=“<PATH>/irods.filetransferplugin.tgz" after_entry="True" const="True" executable="False" untar="True"
    wrapper="False">
                <untar_options dir="irodsplugin" absdir_outattr="IRODS_FILETRANSFERPLUGIN“ cond_attr="TRUE"/>
    </file>
    <file absfname=“<PATH>/irods.pluginsetup.sh" after_entry="True" const="True" executable="True" untar="False"
    wrapper="False">
                <untar_options cond_attr="TRUE"/>
    </file>
</files>
```

Performance Test (I)

Resource Name	iput (1 client)					srm-copy			
	Fermi	Renci	Renci	UNL	UNL (Yaling's node)	Fermi	Renci	UNL	UNL(Yaling's node)
SPRACE	137	133	120	121		117	72	285	
FNAL_FERMIGRID	69	71	72	114		83	91	44	
Firefly	381	528	372	619	535	343	732	38	991
GLOW	123	162	180	303		111	174	87	
CIT_CMS_T2	252	168	196	285	324	150	502	396	2062
Nebraska	66	161	108	83	260	102	42	57	1038
UCSDT2	79	108	107	190	248	79	60	71	2016
Average time in sec to upload 1GB	158	190	165	245	342	141	239	140	1525
Data Rate (MB per sec)	6.48	5.39	6.2	4.17	3	7.26	4.28	7.31	0.67

Performance Test (II)

Resource Name	iput									
	2 clients (1 client per node)		2clients(1 client per node)		3 clients(1 client per node)			6 clients (2 clients per node)		
	Fermi	Renci	Fermi	Renci	Fermi	Renci	UNL	Fermi	Renci	UNL
SPRACE	144	156	142	132	136	149	164	297	306	264
FNAL_FERMIGRID	123	83	76	79	73	74	72	189	184	186
Firefly	191	193	386	388	393	376	372	512	500	447
GLOW	237	93	142	172	224	205	275	212	167	181
CIT_CMS_T2	181	217	263	202	180	160	144	268	225	188
Nebraska	83	74	107	89	90	88	68	157	185	175
UCSDT2	144	85	104	101	137	131	84	235	270	245
Average time in sec to upload 1GB	143		170		171			227		
Data Rate (MB per sec)	7.16		6.02		5.9			4.5		

Performance gradually decreases with increased number of simultaneous clients. It also depends on the SRM Endpoint and client node load at the time of the tests.

Stress Test

Test Num	Test description	File Size	Number of jobs	Number of sites	Average time (sec)	Num of failures	Comments
1	download file from worker nodes (iget)	9.3MB	100	3	0.89	0	File is located at Nebraska, so first is copied to disk cache and then all the clients download the file from there.
2	download file from worker nodes (iget)		500	8	2.22	0	
3	upload file iROD disk cache from worker nodes		100	5	2.23	0	
4	upload file to SE (Nebraska)		100	6	65.2	0	Use iput (lcg-cp) that block until file is uploaded to SE
	Upload file from worker nodes to SE (Nebraska), register file in irods	1GB	100	6	78.59	5	Use iput.py script to copy file directly to SE and then register it in iRODS

Jobs are submitted via glidein pilot using condor transfer plugin. The jobs have been executed on the following Sites: FNAL_DZEROOSG_1, FNAL_DZEROOSG_2, Purdue-RCAC, Purdue-Rossmann, Purdue-Steele, AGLT2_CE_2, GLOW, Tusker, MWT2, VT_OSG. The SRM Endpoint at Nebraska has been used for upload/download files.

Future Work

We have identified the following goals for phase II:

- Modify OSG Frontend configuration
- Upgrade iRODS to 3.1.
- Identify 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.
- Discussed hardware requirement with GOC.