Public storage for the Open Science Grid

T Levshina[1,2] and A Guru[3]

[2] Scientific Computing Division, Fermilab, Batavia, IL 60510, USA
[3] Holland Computer Center, University Nebraska – Lincoln, Lincoln, NE 68588, USA

E-mail: tlevshin@fnal.gov

Abstract. The Open Science Grid infrastructure doesn't provide efficient means to manage public storage offered by participating sites. A Virtual Organization that relies on opportunistic storage has difficulties finding appropriate storage, verifying its availability, and monitoring its utilization. The involvement of the production manager, site administrators and VO support personnel is required to allocate or rescind storage space. One of the main requirements for Public Storage implementation is that it should use SRM or GridFTP protocols to access the Storage Elements provided by the OSG Sites and not put any additional burden on sites. By policy, no new services related to Public Storage can be installed and run on OSG sites. Opportunistic users also have difficulties in accessing the OSG Storage Elements during the execution of jobs. A typical users' data management workflow includes pre-staging common data on sites before a job's execution, then storing for a subsequent download to a local institution the output data produced by a job on a worker node. When the amount of data is significant, the only means to temporarily store the data is to upload it to one of the Storage Elements. In order to do that, a user's job should be aware of the storage location, availability, and free space. After a successful data upload, users must somehow keep track of the data's location for future access. In this presentation we propose solutions for storage management and data handling issues in the OSG. We are investigating the feasibility of using the integrated Rule-Oriented Data System developed at RENCI as a front-end service to the OSG SEs. The current architecture, state of deployment and performance test results will be discussed. We will also provide examples of current usage of the system by beta-users.

1. Introduction

One of the goals of the Open Science Grid (OSG) is to provide Virtual Organizations (VOs) with opportunistic usage of grid resources – one of which is storage. The OSG has initiated a production-scale Opportunistic Storage provisioning and usage on all OSG sites. This paper is focused on describing the current status of storage usage by opportunistic VOs and the efforts devoted to improving accessibility and management of the OSG Public Storage by integrating it with iRODS [1].

[1] To whom any correspondence should be addressed.

Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

2. Data Handling on the OSG

Computation preformed on the grid is often data driven and could be data intensive. Grid jobs usually require one or many input files and produce sizeable output. A user may request to bring data with a job. In that case either the HTCondor [2] data transfer mechanism is used or a job may access data via Squid [3] during the execution on a worker node. Another way to get access to data is to submit a job to a site where data is already located. However this approach requires staging data on a remote site before job execution. Output files created by a job could be either brought back to the submission node by using the HTCondor data transfer mechanism or be uploaded to a local or remote Storage Element (SE).

The HTCondor data transfer mechanism works fine for "small" files. The problem arises when multiple jobs start simultaneous files transfer with file size that exceeds 1GB. Squid can be effective when frequently requested files, such as calibration or configuration data, are used. However Squid is configured with a limited object cache size (250 MB in some cases on the OSG) and hence renders itself of no use for larger file sizes. Squid also doesn't help when each job requires a unique set of files and is not relevant to output data.

Thus for transferring sizable input and output data, the only solution is to use a designated SE.

3. OSG Storage Elements

Typically, there are two types of Storage Elements supported on the OSG sites. The so-called "Classic" SE consists of one or several gridFTP servers and a disk cache that is usually mounted via NFS to worker nodes. In many cases only read access is allowed from a worker node. An environment variable named OSG_DATA is defined on every worker node. It points to the storage area and is defined on every worker node. Not all the sites support this type of SE, and even when supported the size of the storage is limited for non-owner VO (e.g. at Fermilab it is set to 400 GB).

The other type of SE is a cluster of nodes that can be accessed via the Storage Resource Manager (SRM)[4] endpoint. GridFTP servers are used for file transfer in this type of a SE. A user interacts with a SRM SE by issuing file get or put calls. Usually OSG sites have at least one SRM SE. The available space per non-owner VO is negotiable and is measured in TBs.

4. Motivation for public storage

The LHC Experiments are using the Storage Elements of Tier-1, Tier-2 and even Tier-3 sites to pre-stage input data and upload output files from grid jobs. They maintain a file catalog to keep track of data location. There are multiple custom tools used by each LHC experiment to solve data management tasks described above (FTS [5], PhEDEx [6], AliEN [7], LFC [5], SAM-Grid [8], AAA [9]). None of them could satisfy generic data management requirements because they are either tightly coupled with the experiments' internal workflow or require significant investment in hardware, and software support and maintenance.

Most of the OSG sites do not support dynamic storage allocation. The common tools for automatic management of allocated storage do not exist. As a result small VOs have difficulties finding appropriate storage, verifying its availability, and monitoring its utilization. The involvement of a production manager, site administrators, and VO support personnel is required to allocate or rescind storage space.

Our goal is to provide a service (the OSG Public Storage) that will enable small non-LHC VOs whose computation requires "large" data to use OSG sites with less effort. We are looking for a solution that will ease the task of VO data handling by providing quota management, data pre-staging to sites, retrieving output data from sites, and providing metadata catalog. That service would allow the OSG production manager to handle public storage allocation across all the participating sites while imposing minimal burden on sites. This service should simplify discovery of available storage and files location.

We have been searching for an available storage solution that satisfies our requirements that can be integrated with existing OSG middleware, and has strong community support. In this pursuit we are

currently exploring the feasibility of integrating the Integrated Rule-Oriented Data System (iRODS) with the OSG SEs for providing the OSG Public Storage.

5. iRODS Overview

iRODS is developed by the Data Intensive Cyber Environments research group and collaborators. iRODS implements a policy-based data management framework. It allows defining various objects such as resources, collections, and files. Each object has a set of properties (metadata) associated with it. These properties are enforced by polices (set of rules). Rules can trigger a chain of actions (micro-services). A chain of actions may include recovery from failures and notification, provide means to set quota limit, and enforce quota management. Typically, iRODS performs file transfers by using implementation specific protocol to access POSIX compliant resources but it also has a notion of a compound resource that allows integration with non-POSIX storage by an external driver to Mass Storage. The custom driver should implement `put` and `get` methods to transfer an entire file. The compound resource requires a disk cache resource to be configured in the same resource group. In that case, a file transfer is performed in two steps; e.g for upload a file is first staged to a disk cache and then is moved to a remote storage using the *put* driver function.

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 a metadata catalog for users' data collections. iRODS is widely used by the scientific community in fields such as Biology, Environment , Physical Sciences, Geosciences, etc.

iRODS seems to provide multiple benefits if used as a frontend for the OSG Public Storage service:

- It allows a user to pre-stage data to OSG_DATA and SE SRMs without dealing with sites, gathering scattered information about site resources, and worrying about storage location and end path, etc.
- It provides a global namespace that has information about files location, size, etc.
- It manages quota per VO/resource.
- It supports X509 authentication.

Although our deployment has shown promising results, however there are also several drawbacks in our approach:

- Inefficient file transfer since a file pre-staging/download happens in two hops.
- Our deployment cannot fully utilize iRODS features because of the architecture we are using.
- We have to develop and maintain custom scripts.

6. Public Storage Architecture

Currently, the OSG Public Storage service consists of an iCAT enabled iRODS server, the custom universal mass storage plugin that interfaces OSG SEs, and multiple scripts that provide file replication to multiple SEs, resource and user registration. These scripts offer comprehensive information about resource status, metadata, usage and quota. We distinguish between two workflows that are currently supported by our deployment.

Data Pre-staging and Download

A user may want to upload a file from their local machine to a SE. The sequence of events is the following (see Figure 1):

- A user issues a *put* request to iRODS for a specific file and storage resource group.
- A file is transferred to iRODS disk cache.
- The file metadata information is registered in iCAT.
- iRODS server determines that a compound resource has been requested and subsequently calls the universal Mass Storage plugin.
- The plugin selects a remote SE that supports the user's VO that is "*up*" and has enough space within the quota for this file. The plugin then transfers this file using srm (lcg-cp) or gridftp

(globus-url-copy) command to the selected SE. The plugin determines the SE SURL by querying resource metadata in iCAT.

- In the case of a successful transfer the iRODS server registers additional metadata about file location in the iCAT and terminates communication with the client.

A similar set of action is needed for downloading files from remote SE to a local host. It should be noted that in both cases a user doesn't need to know the storage location of the file and has no knowledge about how to find and access the selected SE. If a user wants to replicate the same file on all SEs of the same resource group she can use a custom script to do the replication.

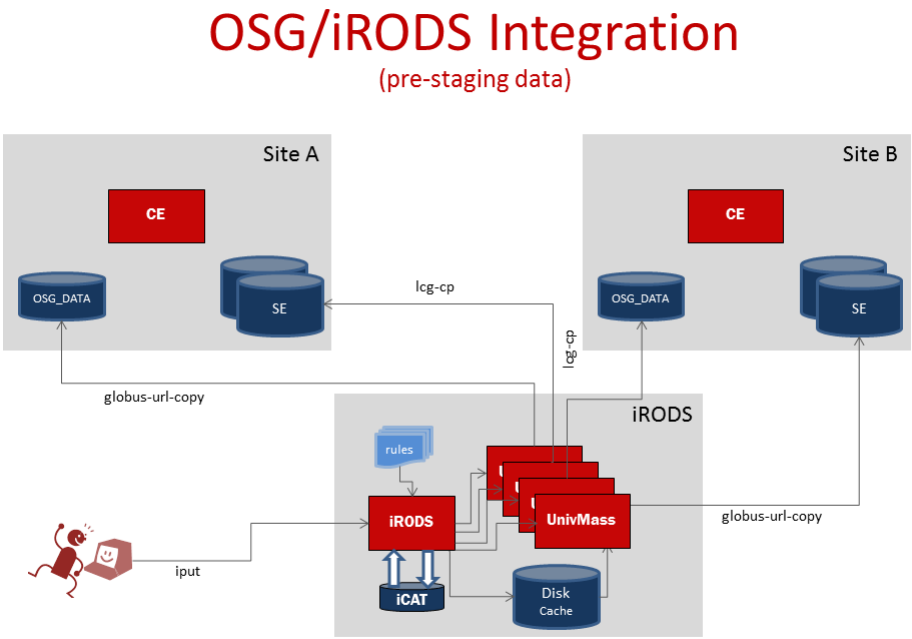

Figure 1. File pre-staging workflow.

Data Access during grid job execution.

The second workflow is used by the user's grid job when it needs to access pre-staged data or upload output files to a SE. In that case the sequence of events is the following (see Figure 2):

- On a grid submission node a user creates an HTCondor job submission file and indicates that the jobs should run on the grid sites where iRODS is installed. In the job's wrapper script a user needs to add a custom *icp* command that has a target and source arguments. In the case when an output file needs to be stored, the source argument will just be a full filename and the target argument will be iRODS server url, e.g.
  idrodse://irodsuser@<irods-host>:<port>?/osg/home/username/<input_file>
- When a job starts execution on a worker node and reaches the point when the file needs to be saved to some SE, the *icp* command contacts the iRODS server and requests information about the most appropriate storage (available, preferably local, with enough space). As soon as the SE is selected *icp* command initiates a direct transfer to a SE. If the transfer succeeds the *icp* command requests file registration in iCAT by issuing the *ireg* command to iRODS.
- In a case where a job needs to get a file that it is already pre-staged at some SE, the *icp* command issues a request to iCAT to get a file location and then downloads the file directly from a SE (preferably local SE).

A user can list all the files and their location and metadata by issuing the *ils* command.

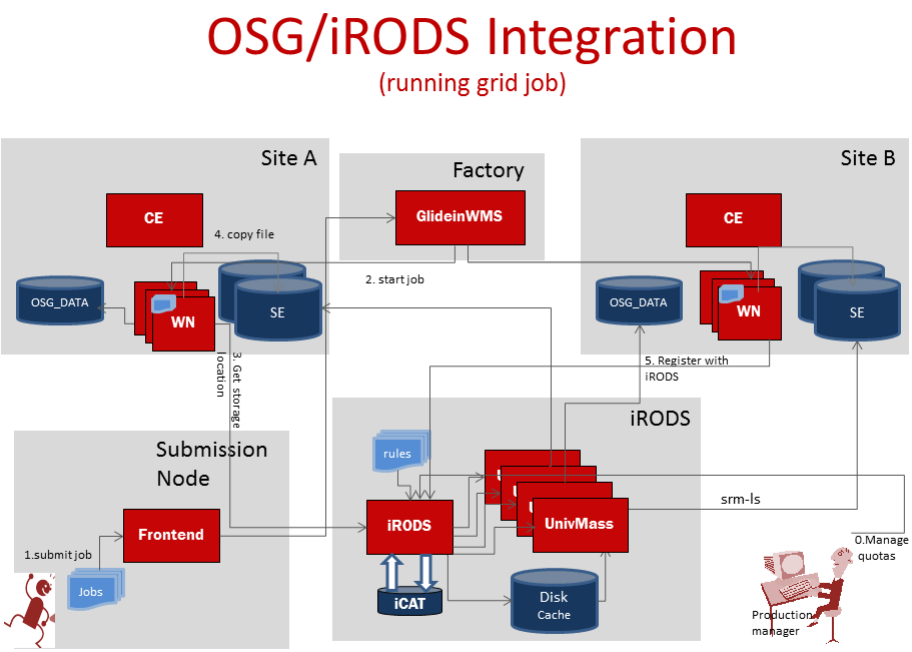

Figure 2. Grid Job Files workflow.

7. Current deployment

The current deployment of the OSG Public Storage service is in the pre-production stage. The iRODS server, the iCAT (postgres) database, and relevant custom scripts are installed on a virtual machine (2 Cores, 4GB memory, 2TB disk cache) at Fermilab. The client's commands are installed on the OSG submission node and are downloaded by glidein pilot to a worker node on glidein startup.

Currently we have defined two resource groups (osgGridFtpGroup and osgSRMGroup with 10 and 23 resources respectively). A resource has the following attributes:

- Quota and Usage per VO/resource.
- Status (up/down).
- Meta data: Supported VO, SE SURL, End path per VO.

We have currently two VOs registered with iRODS (osg and hcc) and 14 "users". A "VO" is defined as iRODS users' group. X509 authentication is used between a user and iRODS , as well as between iRODS and SEs. Multiple certificates can be associated with the same user. A user could be a member of one or more groups.

A user can:

- Pre-stage a file to a specific SE, e.g. *iput –R Nebraska my_file*
- Pre-stage a file to a single SE in some resource group, e.g. *iput –R osgSrmGroup my_file*
- Download a file from SE,e.g. *iget my_file*
- Delete a file from SE, e.g. *irm my_file*
- Replicate a file from one SE to all other available SEs within the same SE group, e.g. *irepl-osg –R osgSrmGroup my_file*
- List file meta data and location information, e.g *ils –l /zone/home/UserName*

The rule that handles quota enforcement is enabled in iRODS core rules. The quota limit change triggers the execution of the rule. It checks if quota is exceeded per group/resource and then may delete files until space utilization is under the limit. It also sends email notifications to the owners or production managers.

A production manager monitors the current space utilization by using the custom script (*ilsresc-osg*) that allows to list resource utilization and quota limits for each VO and each resource that provides storage for that VO.

A production manager needs to be involved in iRODS operation only when:

- A new VO has requested access to iRODS .
- A new resource needs to be registered (could be done by querying BDII).
- Quotas must be modified.
- Change in membership was requested. This request could be executed automatically if we use scripts that sync iRods users with a relevant VOMS service.

8. Use Cases

There are several groups of users that have used this service successfully in pre-production, namely :
SNOWMASS (Simulate hundreds of millions of high-energy proton-proton collisions, which mimic the collisions expected at future hadron colliders) utilizes this service by:

- Pre-staging big files (3 – 15 GB) via iRODS to osgSrmGroup resources
- Using *icp* command to stage files to the worker node
- Using temporary directory defined by ${GLIDEIN_Tmp_Dir} to stage the file at the beginning of a job. All the other jobs that are started by the same glidein will use the file that was downloaded by the first job.

EIC (Electron Ion Collider at BNL: Modeling the performance and optimizing the design) utilizes this service using two patterns:

- Pattern A: Stage files (1 GB) to osgGridFtp resources; run on selected sites, and use *cp* command to copy files from $OSG_DATA to local space.
- Pattern B: Stage files to osgSrmGroup resources; run on all sites, and use *icp* to stage file either from local or remote storage.

DetectorDesign (Medical Imaging, University of New Mexico: Investigating how different simulated SPECT system geometries can affect reconstructed images) utilizes this service by:

- Submitting jobs on limited set of sites.
- Uploading output files to local/remote storage using *icp.*
- Downloading all the files to the local machine via iRODS.

9. Issues

The main issue we have encountered so far is that iRODS doesn't provide any kind of throttling. We have to control the number of simultaneously executed *srm* commands initiated on iRODS machine via the Universal Mass Storage plugin to remote SEs. In order to achieve this we implemented a throttling daemon that controls the number of simultaneously executed srm commands. On startup an instance of the Universal Mass Storage plugin sends a request to a throttling daemon and continues execution only when the daemon sends confirmation back.

10. Conclusions

The OSG still doesn't have a generic approach for public storage. A pressing need to provide data handling solution for small VOs is mounting.

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 action is required from sites after the 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. A user can pre-stage data to the registered resources (SEs).

We need to perform more stress and load tests before we can move to production deployment and offer it as a common solution for the OSG small VOs.

Acknowledgments
Many thanks to Gabriele Garzoglio, Chander Sehgal, and Brian Bockelman for the productive discussions related to the design and implementation of the OSG Public Storage.

Fermilab is operated by the Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the United States Department of Energy.

References

[1] Rajasekar A et all 2010 *iRODS Primer: Integrated Rule-Oriented Data System. Synthesis Lectures on Information Concepts, Retrieval, and Services (*Morgan and Claypool)
[2] Thain D, Tannenbaum T and Livny 2005 Distributed computing in practice: the Condor experience *Concurrency and Computation: Practice and Experience* **17 2-4** 323
[3] Saini K 2011 *Squid Proxy Server 3.1: Beginner's Guide* (Packt)
[4] Donno et all 2008 Storage Resource Manager version 2.2: Design, implementation, and testing experience *J.Phys.: Conf.Ser.* **119** 062028
[5] Frohner A et al 2010 *Data management in EGEE J. Phys.: Conf. Ser.* **219** 062012
[6] Egeland R, Widish T, Huang C 2010 PhEDEx Data Service *J. Phys.: Conf. Ser.* **219** 062010
[7] Buncic P, Peters A J,Saiz P and Grosse-Oetringhaus J F 2004 The architecture of the AliEn system, *CHEP 2004*, Interlaken, Switzerland 440
[8] Illingworth, R 2013 The Fermilab SAM data handling system at the intensity frontier *in these proceedings*
[9] L Bauerdick et al 2012 Using xrootd to federate regional storage *J. Phys.: Conf. Ser.* **396** 042009