OASIS: a data and software distribution service for Open Science Grid

Journal of Physics: Conference Series **513** (2014) 032013 doi:10.1088/1742-6596/513/3/032013

OASIS: a data and software distribution service for Open Science Grid

B. Bockelman[1], J. Caballero Bejar[2], J. De Stefano[2], J. Hover [2], R. Quick [3], S. Teige[3]

[1] University of Nebraska-Lincoln, Lincoln, NE 68588, USA
[2] Brookhaven National Laboratory, PO BOX 5000 Upton, NY 11973, USA
[3] Indiana University, Bloomington, IN 47404, USA

E-mail: jcaballero@bnl.gov

Abstract. The Open Science Grid encourages the concept of software portability: a user's scientific application should be able to run at as many sites as possible. It is necessary to provide a mechanism for OSG Virtual Organizations to install software at sites. Since its initial release, the OSG Compute Element has provided an application software installation directory to Virtual Organizations, where they can create their own sub-directory, install software into that sub-directory, and have the directory shared on the worker nodes at that site.

The current model has shortcomings with regard to permissions, policies, versioning, and the lack of a unified, collective procedure or toolset for deploying software across all sites. Therefore, a new mechanism for data and software distributing is desirable.

The architecture for the OSG Application Software Installation Service (OASIS) is a server-client model: the software and data are installed only once in a single place, and are automatically distributed to all client sites simultaneously.

Central file distribution offers other advantages, including server-side authentication and authorization, activity records, quota management, data validation and inspection, and well-defined versioning and deletion policies.

The architecture, as well as a complete analysis of the current implementation, will be described in this paper.

1. Introduction

The Open Science Grid (OSG) [1] has always encouraged the concept of software portability; a user's scientific application should be able to run at as many sites as possible. Traditionally, this was normally accomplished by compiling the software into a single static binary, or distributing all the dependencies in a tarball downloaded by each job. However, the concept of portability runs against the current Linux software distribution philosophy, and becomes increasingly difficult to achieve as the size of a scientists software stack increases. At this point, portability is a barrier adoption of the OSG and it is not pragmatic to provide software packaging assistance to every OSG Virtual Organization (VO).

Accordingly, it is necessary to provide a mechanism for OSG VOs to pre-install software at the sites where they run. Since its first release, the OSG Compute Element (CE) has provided a directory to VOs, referred to as "OSG APP" (for the job environment variable, $OSG_APP, that points to its runtime location), that can be used for application software installations. The

VO assumes it can create a sub-directory that it owns, install its software into the sub-directory, and have the directory shared on the worker nodes for a site. The OSG provides guidelines for the size and UNIX permissions of the $OSG_APP directory.

The $OSG_APP model has been shown to have a few issues:

- It provides no enforcement or allocation mechanism. A single *badly-behaved* VO can utilize all the space available, denying service to other *good* VOs. When this happens, the local site has to make decisions about how to clean up the shared space. In other words, no quota policies can be enforced.
- The sites likely have an internal prioritization for a few VOs, or even just one.
- It may be implemented inconsistently across sites. Some sites prefer VOs do the installation from the worker nodes; others have $OSG_APP read-only from the worker nodes. VOs typically have to learn the idiosyncrasies site-by-site, and there is little sharing between VOs. Also, when a new site comes to OSG, the whole software package has to be installed for all VOs the site intends to support.
- Common software is often installed in multiple places.
- Unless sites provide for dedicated Worker Nodes (WN), or specific batch policies, installation jobs may need to wait in a queue.
- OSG provides no toolset for distributing software. Sites provide a writable directory, but VOs may want higher level concepts such as "copy this file to all accessible sites".

A new mechanism, allowing the distribution of the new software after a single installation, is desirable.

2. Architecture

The architecture for OASIS is a server-client model. Installation is performed once, at a central place or 'server', or a reduced number of places, and the software is visible automatically to all grid sites or 'clients'.

Login on the server host, is restricted to a very limited set of users, typically VOs software managers. Once authorized users are granted permissions to interact on the server side, they can run the VO sofware installation tasks and create new content on a dedicated disk space. This source area on the server is otherwise immutable and does not accept any other type of input. Once installation is done, OASIS provides for robust and secure mechanisms to perform the proper distribution of the new content to all grid sites.

For this server-client architecture, OASIS has to rely on some existing file distribution tool. The current choice for the underlying technology is CVMFS [2].

3. Server

The current deployment of the OASIS service provides for a single server host, deployed and maintained at the OSG Grid Operation Center (GOC). Adding new OASIS servers is a transparent process from the client point of view.

3.1. Login

Two different mechanisms are envisioned for users to interact with OASIS server in order to install and deploy new content.

The first mechanism, already deployed, allows for SSH [3] login with GSI [4] authentication and authorization. Users can login interactively to a user interface host, once their GSI credentials have been accepted. Only those previously accepted and registered credentials will be granted access to the server host. Once logged into the interface host, users can run the installation job and invoke directly an OASIS command line tool to trigger the publication of the new content.

The second mechanism will provide access to the server services via a gatekeeper, as Figure 1 shows. In this case, access will be granted only to those credentials carrying a pre-specified VOMS [5] attribute. Once the job has been accepted, it is run on one host in a local batch queue within the OASIS server. The batch queue is managed by HTCondor [6], and the job runs within an OASIS-specific job wrapper. This allows OASIS to control all aspects of handling the installation process.

When a new VO is created, the authentication and authorization mechanism in place at the server adds the appropriate new accounts for that VO's authorized managers. To guarantee VO isolation, each VO manager's credentials are mapped to a different UNIX account.

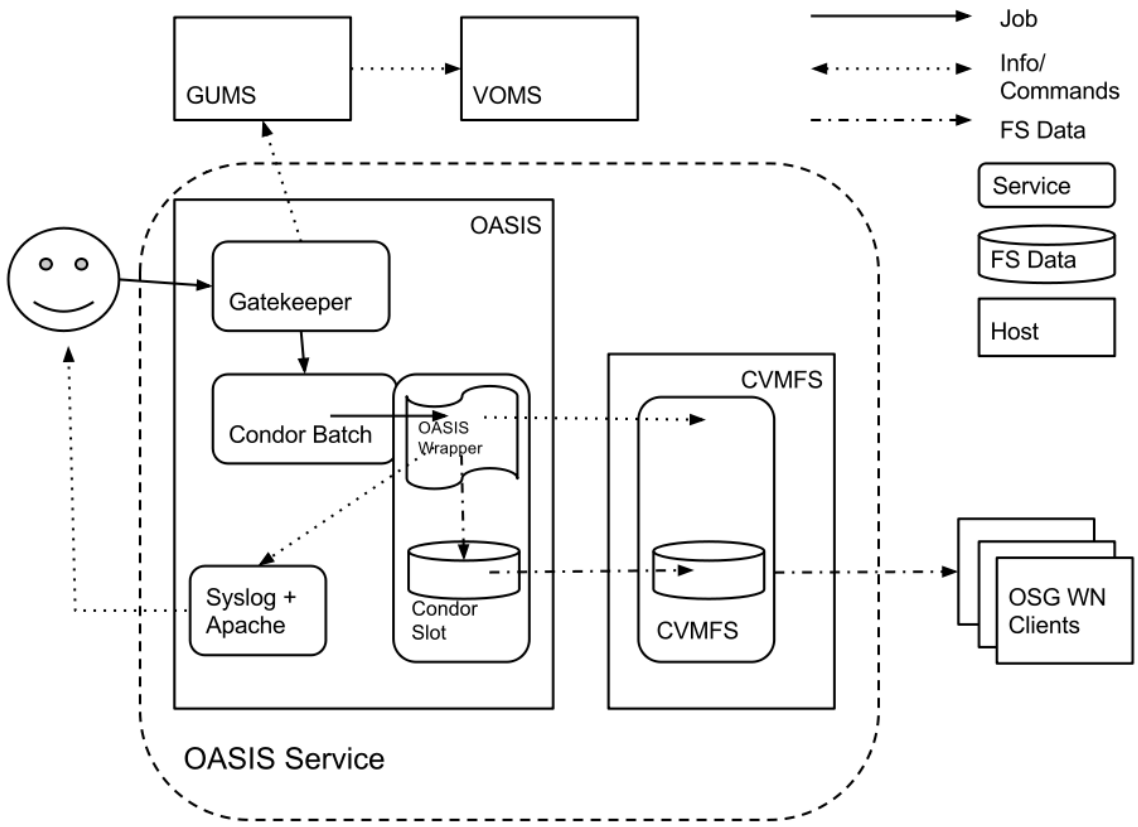

Figure 1. CE-based server diagram

3.2. VO software installation
In the case of direct ssh access, users can run their sofware installation applications, and invoke the OASIS CLI tools. When access is done via the gatekeeper, the user payloads are executed on a local batch queue managed by HTCondor. A dedicated OASIS-specific HTCondor job wrapper can perform some preparatory steps, run the user installation payload, invoke the internal OASIS CLI, and perform some post-install tasks. All these steps are transparent to the end users.

VO installation jobs write the new content into a VO scratch area (indicated by the traditional $OSG_APP environment variable). After installation is completed, the OASIS publishing script is invoked. This publishing program performs three tasks, in this order:

(i) runs a set of probes to validate the new content,
(ii) if, and only if, all probes are passed successfully, transfers the content into the distribution tool source tree,
(iii) calls for the distribution tool publication command.

The design of the OASIS server services, based on a plugin-in architecure, allows for flexibility and modularity on all three steps. Each probe is implemented as a separate component, can be enabled or disabled on a VO by VO basis, and can be written to accept arbitrary input options as appropriate. The underlying technology for file distribution/publication could be switched from one implementation to another in a transparent way to the users, e.g. in theory the role played by CVMFS could be played by some other distributed filesystem.

All activities performed by the users at the server side can be recorded for bookkeeping, troubleshooting, and accountability. Current status of installation tasks, as well as historical summaries, can be displayed in a web monitor.

3.3. Probes
Prior to the publication of new content, a series of probes confirm that all policies are fulfilled. These policies are both generic and VO-specific. Each VO decides which probes to run, which configuration, and whether they are just informative or critical. If any of the critical probes fail, then the publication process is aborted. Some examples of probes are listed in table 1.

All these probes may accept input values to customize their behavior, e.g. to setup exceptions, to only look at a given directory path or file, to choose between just raising a warning or aborting the installation in case of failure.

4. Client
The software and data deployed at the server are automatically visible at client sites. In order to guarantee authenticity of the content at the clients, some security mechanism must be in place in the communication between clients and server. Using CVMFS as a content distribution tool, this can be done using the same X.509 [7] standard for a Public Key Infrastructure (PKI) already being used regularly on OSG.

It is desirable for the sites to run some type of replica (or cache) service, to prevent all client hosts from reading content directly from the server. Clients then get the content from a replica site, preferably through one or more Squid caches [8]. CVMFS already implements this strategy, based on Squid, in a three layer architecture.

Probe	Scope	Description
Overquota	OASIS	Ensures the VO does not use more space than allowed.
Filesize	OASIS, VO	Ensures every new file is neither smaller than a *minsize* nor larger than a *maxsize*.
NoDeletion	VO	Guarantees that no previous file has been deleted.
NoReWrite	VO	Guarantees that no previous file has been modified.
NoTarball	OASIS	Prevents from a tarball file being distributed.
CatalogSize	OASIS	Checks the total number of files and the size of the resulting CvmFS catalog. If needed, it can take corrective actions such as forcing the splitting of the catalog into several subcatalogs.
Relocatable	OASIS	Ensures every binary being distributed is relocatable, and not linked against files whose path will be missing on the client nodes.

Table 1. Example of OASIS probes

The new content being distributed is allocated on a specific filesystem, with a meaningful name, i.e.

```
/oasis/<voname>/
```

However, the variable $OSG_APP still remains for VOs not interested on using OASIS.

For those VOs migrating to OASIS service that have, for historical reasons, the string $OSG_APP hardcoded in their jobs software, a symlink is needed:

```
$OSG_APP/<voname> -> /oasis/<voname>/
```

5. Summary

At the most abstract level, the goal of the OASIS system is to provide a scalable, safe way for VO software managers to add content to a globally accessible filesystem. This allows VOs to avoid the onerous task of managing their software on a site by site basis.

VOs must be protected from each other such that actions by one VO software manager cannot affect other VOs. VO software managers must be protected from themselves, in the sense of preventing mistakes that might result due to lack of familiarity with the underlying distributed filesystem.

Lastly, the content installation system should allow the replacement of the underlying distribution implementation without requiring VOs to make changes to their practices or procedures.

We believe this design sketch for OASIS can satisfy all these goals well.

Acknowledgments

The authors would like to thank the CVMFS developers for their constant support. We are also grateful to the site administrators and experts who have helped with technical questions and made possible the development of this work.

References

[1] Pordes R e a 2007 *J. Phys.: Conf. Ser.* **78**
[2] Blomer J, Aguado-Sanchez C, Buncic P and Harutyunyan A 2011 Distributing LHC application software and conditions databases using the CernVM file system *J. Phys.: Conf. Series* **331**
[3] Ylonen T e a, 2006 The Secure Shell (SSH) Authentication Protocol *Network Working Group RFC 4252*
[4] URL `http://www.globus.org/security/overview.html`
[5] URL `http://www.globus.org/grid_software/security/voms.php`
[6] Thain D, Tannenbaum T and Livny M 2005 Distributed Computing in Practice: The Condor Experience *Concurrency and Computation: Practice and Experience* **17, No. 2-4** 323-56
[7] Adams C and Farrell S, 1999 Internet X.509 Public Key Infrastructure: Certificate Management Protocols *Network Working Group RFC 2510*
[8] URL `http://www.squid-cache.org/`