Open Science Grid

Document Name	**Open Science Grid Architecture**
Version	
Date last updated	Feb 10th 2009
OSG Activity	Blueprint

The Architecture of the Open Science Grid summarizes, outlines a future direction for and provides pointers to more information about, the Business model, the Information model, the Application architectures, and the Technology architectures of the OSG.

V0.1	First pass based on face to face meeting and inputs since then

1. Introduction

1. Goals and Scope

The Open Science Grid (OSG) represents a distributed facility that provides access to computing and storage resources at various locations in the US and abroad. This document describes the enterprise architecture of the OSG. The primary purpose of describing the architecture of an enterprise is to improve the effectiveness and/or efficiency of the business itself with minimal impact on the operations or agreed-upon SLAs.

There are two goals for this document: to capture the current OSG architecture and to define an architectural direction for the future to meet needs as known today of current and potential stakeholders.

The scope of the enterprise architecture practice is the description of the whole enterprise from different perspectives: Business (including the operating model and processes), Information (including the organization's meta-data), Applications, and Technology (including traditional hardware and software architecture).

This document should not be read standalone. In particular more details are given in documents such as the: Introduction to OSG, Charter of the Open Science Grid, and OSG Blueprint (see table below for references). This document is also a snapshot with annual updates. The details in the updated documents from the document repository and from the collaborative TWiki supersede it. You will need to read the above documents, as well as related pages on the collaborative TWiki in order to understand the details of the OSG current practices and its status.

2. Enterprise Architecture

We are using the following rules (with more detail in Section 8) to guide us in the development of this document:

- Describing a "target" or "future state" goal is as important as a detailed description of the status quo.
- Architecture focus is to iterate over solution descriptions (end-to-end) rather than components.
- Importance of engaging individuals and their interactions in the architecture should be explained (not clear how to do this yet).

We describe the OSG architecture from four enterprise viewpoints:

- Business
- Information
- Applications
- Technology

3. Related OSG Documents

This is not the first document about the OSG enterprise architecture, a list of previous work related to different functions is provided in this section.

Business	Introduction to	http://osg-

	OSG	docdb.opensciencegrid.org/0008/000839/002/OSG%20Intro%20v11.pdf
Business	Charter of the Open Science Grid	http://osg-docdb.opensciencegrid.org/0000/000025/004/Charter.pdf
Technology	OSG Blueprint	http://osg-docdb.opensciencegrid.org/cgi-bin/RetrieveFile?docid=18&extension=pdf
Information	OSG Core Assets	http://osg-docdb.opensciencegrid.org/0007/000762/004/OSG_Core_Assets_v_3.xls
Business	The Value of the OSG	http://osg-docdb.opensciencegrid.org/0008/000813/001/OSG%20Value%20Proposition-V1.0.pdf

4. The OSG

The OSG mission is defined at http://www.opensciencegrid.org/About/Learn_About_Us/Our_Mission:

> *The Open Science Grid aims to promote discovery and collaboration in data-intensive research by providing a computing facility and services that integrate distributed, reliable and shared resources to support computation at all scales.*

The goal is to provide a plan for extending this mission to:

> *The Open Science Grid aims to promote discovery and collaboration in data-intensive research by providing computing services, software and support that integrate cross-domain, self-managed, nationally distributed cyber-infrastructures that bring together campus, community, national and commercial resources to meet the needs of communities at all scales.*

The architecture and targets of the OSG are driven by its participating virtual organizations (VOs) – current and potential. IN OSG, a VO refers primarily to a collection of people and also includes the group's computing/storage resources and services. The (three) major VOs that drive the OSG at the time of this writing are the US LHC Collaborations (US ATLAS and US CMS) and LIGO. Currently:

- The US LHC Collaborations depend on and contribute to the OSG as part of their distributed computing facilities in the US. US ATLAS and US CMS make resources, services and software accessible through the OSG. They implement agreed upon policies for the opportunistic, shared use of these resources by other research communities. They implement policies to bring and share software and services to the common good.
- LIGO depends on the OSG software stack and experienced collaboration to provide support. The LIGO scientific collaboration federates it's distributed community facility with, as well as contributes resources accessible through, the OSG.

OSG itself includes VOs that additionally drive the architecture and targets:

- The Engagement community brings a diverse set of users and communities to use the OSG who do not, in general, have their own resources to offer. As these users and communities become self-sufficient they may transition to full stakeholders in the OSG.

- The Education community is a small community whose focus is on education and training of students and the workforce. This community is one of the drivers of documentation and training activities.
- The OSG community provides an umbrella for OSG staff and any member of the OSG who wants to test, run existing applications as individuals, or provide services for other communities to share.

Our target is to increase the number of major VOs depending on the OSG, the value of the common software and services provided to an expanded community.

2. Business Architecture

The OSG project is built and operated by the OSG Consortium - many autonomous organizations (some of which are virtual[1]). Consortium members contribute effort and resources to meet the distributed computing infrastructure needs of the members. Research communities, computing resource providers, software development projects, educators and students are members of the Consortium. Membership is based on registration.

The goal in any enterprise architecture is to bring the business side and the technical side together, so that both work together towards the same targets. The business aspects of the OSG covered here are:

1. Enterprise goals and operating model.
2. Capabilities and capacities.
3. Organization and organization cycles.
4. Processes and policies.
5. Suppliers of solutions, technologies and information.

1. Enterprise Goals and Operating Model

The current goal is to meet the mission for our stakeholders and be regarded as the premier distributed computing collaboration in the US.

The operating model is that of a grass roots collaboration where specifically funded efforts are combined with unfunded, or parallel funded, contributions to make a more comprehensive whole. Teams of experts and non-experts collaborate to solve the problems and moving the enterprise forward, combined with specific activities directly managed and funded by the OSG. OSG also provides various services to enable these collaborations as well as a Grid Operation Center (GOC) available during all US business hours.

The target is to provide wider and deeper value in the quality and usability of the delivered services and expertise. The envisaged steps towards this target are as follow:

- Consolidation of the Consortiums scope and deliverables to provide value at different levels of member involvement. This might include only use of the software, only participation in operational support, etc.
- Extension of the OSG VO to directly own/lease resources as well as services and software.
- Extension of the OSG VO to include an allocation policy for use of resources owned or leased by and/or opportunistically available to the OSG project.
- Inclusion of WLCG/LHC Applications as an OSG “Maintenance and Operations” area.
- At least one additional science community area based on user need and engagement.
- Improvement of effectiveness and reach of the organization through candidate activities such as:
    - Increased integration and commonality of infrastructure, services, software and support across one or more of the members.

1

Wikipedia: independent organizations that share resources to achieve their goals; see virtual enterprise.

    - Mutually beneficial satellite and/or directly funded joint projects between multiple members of the OSG Consortium and TeraGrid.
    - Mutually beneficial cooperative satellite and/or directly funded joint projects between multiple members of the OSG Consortium and equivalent European projects.
    - Appropriate out-sourcing to satellite or partner projects of particular technical areas.
    - Joint work with TeraGrid, EGI and other CI projects where this provides value.
- Increased attention to the service and software deployment, testing and quality assurance lifecycle.
- Events and activities for OSG staff to increase their technical and distributed computing system skills and knowledge.
- Events and activities for students and new recruits to increase their technical and distributed computing systems skills and knowledge.

2. Capabilities and Capacities

With the enlargement of OSG, the number of resources and the complexity of services is expected to increase. These are some of the elements that have to addressed with higher priority or at least well understood.

2.1. Future Goals

Our future business targets for the OSG are to:

- Increase the performance, scope and capabilities of the infrastructure to meet the needs of the Consortium members up to 2015 through directly funded efforts as well as consortium, satellite and partner contributions.
- Reduce the human user, administration and maintenance loads for using and supporting the infrastructure through simplified and improved technologies, partnerships with like organizations, and increased sharing in services, software and knowledge.
- Extend the reach of OSG to new resources, new communities and new workforce through directly funded efforts as well as consortium, satellite and partner contributions.

2.2. Configuration Management and Native Packaging

These two technical design decisions can complement one another, since native packaging systems provide an integrated post-install configuration hook (rpm-post script in RPMs, and a similar mechanism in DPGK).

There are two ways in which the situation with OSG software is different.

First, while most typical Linux software only needs to know about the host upon which it is being installed in order to be configured properly, typical OSG software needs information about the distributed environment in which it is going to work. It needs information about the site layout (storage, Gratia collector, batch system, GUMS host, etc.) and grid-level configuration (OSG Gratia collector, BDII, etc).

Second, in the context of a Linux distribution, the post-install (or post-upgrade) package script is a one-time mechanism. It is only triggered on install--normally there is no ongoing configuration mechanism. Typical Linux distribution vendors assume manual administration after install. But for OSG site administrators, it would be very useful for OSG software to be fully integrated with an ongoing configuration management mechanism, both at install time and afterwards.

2.3. Scalability & Performance Targets

2010	Support for integration of Virtual Machine across multiple sites for production applications and/or services (STAR, possibly CMS) Support for >100,000 jobs/day/VO (CMS, ATLAS)
2011	Support for and integration of Compute Elements with >50,000 cores (CMS). TBA

2.4. Usability and Robustness Targets

2010	Support VO/customer based web portal job submission and data management (SBGrid, CompbioGrid)

Enable high availability solutions and expansion in automated load balancing across multiple CEs and SEs per site without publishing new information to OSG (registration or information services (ATLAS, SBGrid).

2011 Improve Site level information, monitoring and troubleshooting capabilities. Address OSG Principle that Sites and Applications can support OSG work without connection to the wide area networks.

Improve Site level information, monitoring and troubleshooting capabilities. Address OSG Principle that Sites and Applications can support OSG work without connection to the wide area networks.

Improve ability for Federated infrastructures using a sub-set of OSG services (Campus Grids).

TBA

2.5. Feature requests

- The experience with the current production system has lead to several feature requests that are needed to support current and envisaged growth in the OSG:
- Improved Gatekeeper for job execution - evaluation of GT5 and/or Cream deployment
- Configuration Management across locally and remotely distributed services, versioning, repositories, provenance and validation.
- Improved robustness, usability and integration across web and grid service security infrastructures.
- Improved support for collaborative analysis in environments natural to the research communities, including user access to shared storage, grid level diagnostic tools, dynamic (short and long lived) collaborative activities within communities, scalability of non-organized usage.
- Integration of commercial and scientific clouds into the OSG infrastructure and community use, providing transparency at the application layer and standard interfaces to resources.
- Active management of shared capacity, utilization planning, accounting and reporting, and change.
- End-to-end data management challenges in light of advanced networks, including dynamic circuit reservations and dynamic data placement.
- Better support for parallel jobs.
- Increased effort to enable and to support Campus Infrastructures, including: providing local experts to help scientists get started; train system administrators to make more effective transfer of expert knowledge and management of resources; show value of participation in a national community which may help leverage local investments, increase revenue and diversify funding sources.

2.6. Long Term Evolution

For the long term evolution, several directions need to be considered. Some of these that have been already triggered and identified based on the current requests are as follows:

- Cloud Interfaces to sites: the cloud paradigm has received a lot of attention and several technologies developed. In order to support an extended community of users, cloud interfaces and technology adoption are a sound choice.
- Global file systems with caching:
- Other CE paradigms: the current solution for distributed access to resources is based either on Globus technology (GRAM) or the LCG software stack (CREAM CE). So far neither of these solutions have proved the needed scalability requirements. Thus, other technologies and solutions have to be pursued, investigated and adopted if feasible.

- Data and Job co-scheduling: this is an important element when we move to the scale of a grid like OSG. The data pipes might be insufficient to support all the required movements and in the same time previous research in this area has shown that data scheduling in advance has impact on job scheduling
- Campus and Regional federations: this is an extension of the work already pursued in the LCG context where sites are grouped by regions with various operation centers.
- Laptops and mobile computing: these represent an important element to consider – job and data management should not require complex systems while with the advent of more power for these devices they can even sustain adequate computations.
- Portals and visualization:
- Ad-hoc collaborative groups:
- Management of Ids:
- International partnerships and reach:
- Data intensive science:
- Usability and troubleshooting:

3. Organization and Organization Cycles

The OSG Council provides the governance of the OSG. Membership of the Council follows the agreed upon by-laws, which provide for direct representation of the major contributors and "at-large" members for the remaining members of the Consortium. The work of the Consortium is achieved through:

a) DOE/NSF funded OSG Project consisting of ~33 FTEs funded to support specific activities delivering to the OSG mission until mid-2011. The work of these 33 FTEs is agreed to in annual Statements of Work with their institutions
b) Contributions from Consortium members of the computing resources, user applications, shared services, software and support.
c) Partnership agreements where the partner organization and OSG agree to a set of activities and contributions.
d) Contributions from Satellite Projects, independently funded projects of several specific kinds[2].

The Consortium and Project interact as defined by the OSG Management Plan and Cooperative Agreement with the DOE and NSF program offices.

The deliverables of the Consortium are defined through decisions of the Council, under the management of the Executive Team, executed by the Executive Board. The deliverables of the OSG Project are defined and monitored through an annual program of work agreed to by the Executive Board and Council. The Organizations is described at http://www.opensciencegrid.org/About/Learn_About_Us/OSG_Organization.

Currently the scope of the OSG project covers the activity area defined below:

Maintenance and Operations:	Extensions:
Software Tools Group	Scalability, Reliability and Usability

2 OSG Interface to Satellite Proposals/Projects….

Production Coordination
Biology & Biomedical
LIGO Applications
Software
Operations
Integration & Sites
VOs
Engagement
Campus Grids
Security
Training & Content Management
Work Load Management
Internet2 Network Monitoring

Administration:
Metrics & Measurements
Communications & Education
Project Management

Software Support:
Condor (FY10 only)

3.1. Future Organizational Targets

As already mentioned, there are three main VOs that participate a large part of the OSG resources and also OSG's VOs. Based on the current experience, our future organizational goals are to:

- increase the number of organizations participating in the Consortium and with appropriate representation on the Council.
- increase the effectiveness of the Council / Project interactions.
- grow the staff technical and organizational expertise and skills.
- encourage new recruits to help support and evolve the infrastructure.

The steps towards the organization targets include:

- Providing direct administrative and technical effort in support of the Council Co-chairs
- Addressing issues of end-to-end CI to have a community-wide ownership of architecture
- Addressing the non-technical challenges.

4. Processes and Policies

OSG policies are listed at http://www.opensciencegrid.org/docdb_dashboard/index.php and cover security, expectations of members and member organizations providing services, resources and software to the OSG, and commitments for the services and software provided by the Consortium.

Operational, release, response, planning, and service processes are developed as needed, the responsibility of an assigned area coordinator of member of the Executive Team, and maintained on the collaborative workspace Twiki.

Future Target:

- Improve maturity of software deployment through processes that include risk assessment; Integration, operations, scalability testing; Security auditing/risk assessment; Error and fault information, translation and propagation; Diagnosis/troubleshooting; Documentation and training; Life-cycle planning

Steps towards the Future Target:

- Sponsor activities for co-design, review and support for end-to-end software solutions.
- Extend the Virtual Data Toolkit to include support for 2 software packages whose current developers have left and pilot the process for support across the communities who depend on it.
- Publish results from risk assessment, evaluation of use usability, design, performance, security, re-training.

- Identify, discuss and motivate functionality to move out of narrower application domain into commons (in the same community, outside the community)
- Continue VOSS survey of software project characteristics in the field covering one or more of: Individual and collective motivation; Organizational structure, scope, and scaling; Organizational life cycles; Production and innovation; Management, Governance, and Leadership; Measurement and assessment; Units and frameworks of analysis-both social and technical; Comparative performance:
- Address realities that support inevitably includes evolution (response to faults, extension to new requirements).
- Forum for principles of what components should include/exclude. As important to understand what should Not be in software as what should be in. How we decide where the boundaries are. How we minimize complexity. Where there are and recommendations to deal with overlap & gaps. How and how long to accommodate legacy.
- Planning for, identification of need for next generation of fabric & infrastructure.
- Awareness of software aging and how to measure and evaluate: "Span of changes," increases over time (changes to mature s/w can be more risky..); "Breakdown of modularity; "Fault potential".
- Connection between data preservation and sustaining software. Address cost/benefit as well as potential of newer initiatives. Understanding of end-of-life – not just "drop dead".
- Strengthen use of "low-overhead" agreements.

5. Suppliers of Solutions, Technologies and Information

A well-defined set of solutions, technologies and information are provided by the OSG project itself based on the requests of the stakeholders, expertise and effort available in the project. The model adopted is of ensuring maximal sharing and commonality across multiple science, administrative and user communities.

A larger set of solutions, technologies and information are the responsibility of the research communities, resource and software providers (externally funded) that contribute to the use and capabilities of the end-to-end OSG system. It is required that the technologies be open-source.

The OSG defines its commitments and interfaces to, and expectations from these suppliers.

Future targets:

- Higher or 100% interoperability between among similar software components within OSG
- Dedicated support time from the external components providers through OSG support channels
- Component's performance evaluation by the OSG and documentation
- Support for feature and interface change requests to external components
- Platform oriented packaging as described in the business architecture

Steps towards the future target:

- Connection between component providers and OSG support
- Enhanced tools for request management and interfacing
- External component testing and evaluation on dedicated OSG resources

3. Information Architecture

1. Overview

The OSG Information Architecture describes the data owned by the OSG itself as well as information stored on behalf of the members. OSG keeps information about registered resources, support centers, services, and Virtual Organizations. The information is maintained by OSG Operations and is made available to OSG Staff and members of the Consortium.

OSG keeps its reference documentation close to the source of the data whenever feasible (e.g. VDT documentation as part of the VDT); more general documentation is kept on the main web portal, the Document Repository and the Collaborative Twiki.

2. OSG Information

The up-to-date list of OSG core assets is in http://osg-docdb.opensciencegrid.org/cgi-bin/ShowDocument?docid=762

OSG has the following information repositories with write and read interfaces as shown below:

Item	Write Interface	Read Interface
Open Science Grid public web site http://www.opensciencegrid.org/Site_Map	Xenomedia content management system	url
Official Document Repository http://docdb-v.sourceforge.net/	Mysql inserts	url+ sql
Collaborative twiki	Twiki registration and editing.	url
Software caches	pacman	url + pacman
Mailing lists http://www.opensciencegrid.org/Consortium_Mailing_Lists	Registration on list server	url
Trouble ticket systems See table 3 below.	Mail + web form	url
Information Gateways		
OSG VO Memberships		
ST&E process	?	
Policies	Through twiki and document repository	url
Set of Trust relationships	?	
Registration	Web form (OIM)	PKI security + url
OSG DOEGrids RA https://twiki.grid.iu.edu/twiki/bin/view/OSGRA/	Administration	
VOMS		
CeMON /BDII		

MonaLisa service		
Gratia		
ReSS		
RSV		
Monitoring and measurement website		
GOC GUMS		
CA Distribution		
GIP		
Sharepoint collaborative service		
Matchmaking service		

Ticketing systems:

OSG Ticketing system Users Footprints infrastructure.Training: http://www.grid.iu.edu/docsfiles/docs/Footprints_Training.ppt	Web form at https://ticket.grid.iu.edu/goc/open	url
VDT tickets Uses RT infrastructure. http://vdt.cs.wisc.edu/support/tickets/	Email to vdt-support@opensciencegrid.org	url
ITB tickets Uses Footprints infrastructure. Training: http://www.grid.iu.edu/docsfiles/docs/Footprints_Training.ppt	https://ticket.grid.iu.edu/itb/open	

3. Information Targets

The target information architecture will include information about

- Applications being run on the resources
- Security validation and assessment of the software run and provided on the resources
- Trends in usage by VO and user.
- Data management trends.
- Identity of every user on the resources by means of advanced authentication and authorization mechanisms

3.1. OSG Information Services

Currently OSG uses various tools for information provisioning. The most important ones are:

• MonaLisa Service	Resource Monitoring	WebService
• Gratia	Accounting	html,xml
• RSV	Resource Monitoring	html,
• GOC GUMS	Authorization	xml,WebService
• CA Distribution	RPMs	ftp,http
• GIP	Accounting	Gratia

• Matcmaking Service	Scheduling,Monitoring	ClassAds
• Sharepoint	document exchange	html,xml

3.2. Issues with Information

Besides "Software Caches", the OSG project through VDT provides also " Software Integration ". In the second scope, the "collaborative twiki" can be extended by incorporating forums that link VOs and external or internal packages' developers. One problem raised by different people is that at the current time there are too many redundant monitoring software packages and none of them offers “everything”, while the information should be presented in a form that can be used by automated agents. For example the CMS PhEDEx package offers various monitoring views over various language conversions for the same data based on the http protocol (e.g., perl, json, xml).

Another element that needs improvement is the RA infrastructure with 30 agents. Automated solutions have to be devised such that everything is transparent for the users and the turn-around time small.

The 3 exiting ticketing systems are completely decoupled at this time. The missing communication and synchronization mechanisms impede in many cases the incident management in a coherent way sometimes – the OSG resources represent a large distributed system, not completely isolated instances.

And a big miss is the lack of data monitoring within OSG. VOs cannot build their own infrastructure on anything provided by OSG and they instead rely on their own self-grown solution like PhEDEx and LFC. OSG targets to integrate as much as possible from these lessons and tools within its infrastructure but this needs time, resources and coordination for importing only what is really needed. Abstracting over and deriving the right solutions from VOs ones is hard.

3.3. Information Services - CMS Example

For a better understanding we provide here a snapshot of the CMS information systems. They are categorized upon the acting levels.

VO-level
- Dashboard
    - - Application Execution Results
        - - Interactive&History View
        - - Task Monitoring
        - - Production Monitoring
        - - Site Status Board
    - - Validation in Progress
        - - Task Monitoring
        - - Interactive View
        - - History View
        - - Site Status Board
    - - Old Applications (IO Rate Monitoring)
    - ReadInterface: html

- SiteDB
    - - Site Directory

- Person Directory
- Resource Pledges
- Reports
    - LCG Sitelist
    - Software Install
    - CMS to SAM
    - Naming Conventions, etc

ReadInterface: html,xml

DBS

- search for data sets and individual files at different level (raw to processed)
- search for conditions
- allow users to easily use pusblished data using LFNs (URI)
- interface&API for agent consumption

ReadInterface: xml,WebServices

GUMS

- store and provide the mappings for the Grid ids (DNs) to the local site id (local accounts)

ReadInterface: WebServices

BDii

- site configuration monitoring and publishing
- heirarchic implementation

ReadInterface: ldap, html

PhEDEx Web

- make subscriptions for datasets
- approve them
- retrieve log transfers

ReadInterface: API(webservice), http+(xml or perlObjects or sgml)

Site-level

Pakiti - keeps track of the installed RPMs
JobMon- keeps track of jobs and logs
JobView - keeps track of farm usage
CondorView - keeps tracks of farm usage
SRMWatch/dCache monitors
- keeps track of scheduled transfers

Gratia- provides aggregated reports about data transfers
PhEDEx / FTS- schedules and tracks the transfers

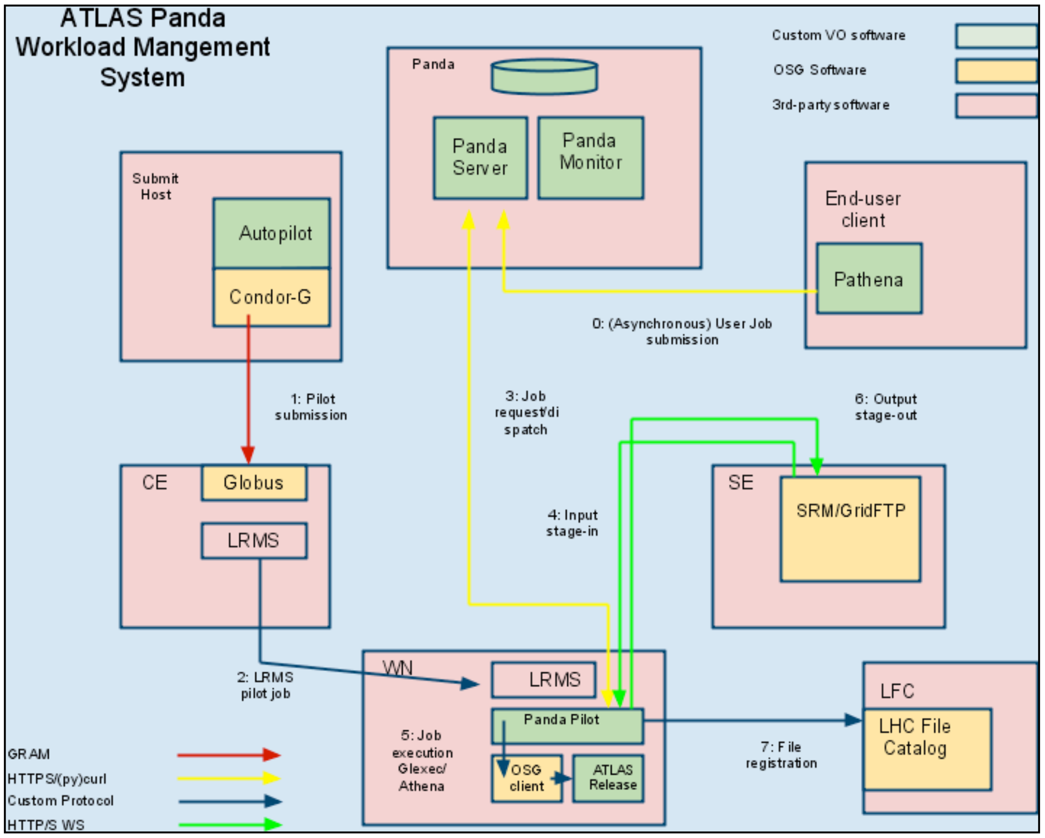
4. Application Architecture

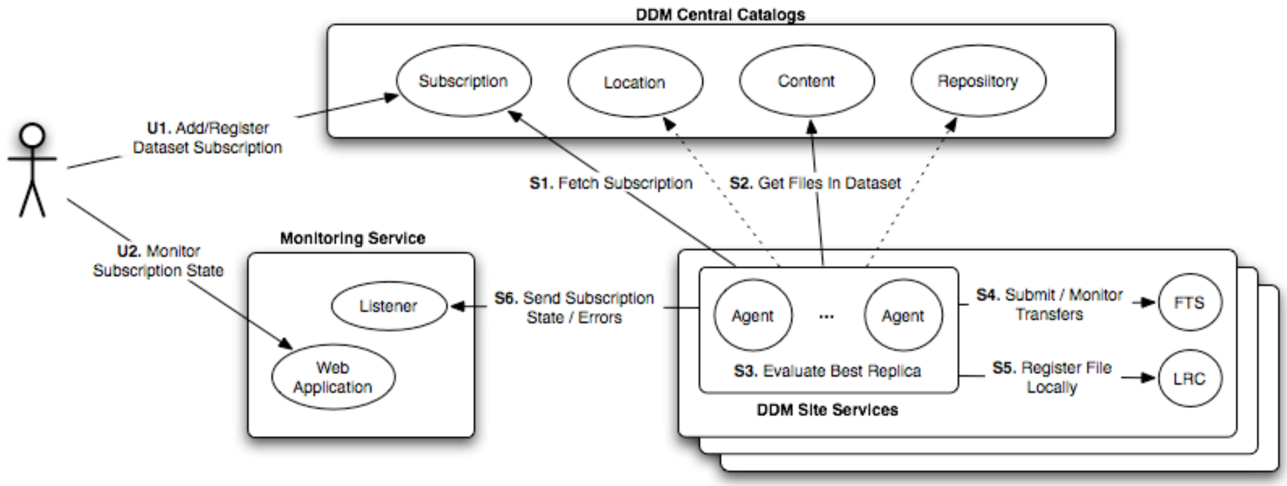
Each community or VO owns its end-to-end application architectures. For many VOs the OSG services provide a "thin" horizontal layer between the client and server services the VO runs and uses. For other VOs OSG services provide a full end-to-end user system tailored, with help from the OSG itself, to meet their application needs.

We document here some of the larger VO architectures as representative.

1. ATLAS

The ATLAS Workload Management system is presented in the following image, while each component is described next. A DDM diagram for ATLAS is also presented next, while the ATLAS Resource Security is identical to the CMS model and thus presented only once in the next CMS section.




DDM Central Catalogs
Subscription
Location
Content
Repository
U1. Add/Register Dataset Subscription
S1. Fetch Subscription
S2. Get Files In Dataset
U2. Monitor Subscription State
Monitoring Service
Listener
Web Application
S6. Send Subscription State / Errors
Agent
...
Agent
S3. Evaluate Best Replica
DDM Site Services
S4. Submit / Monitor Transfers
FTS
S5. Register File Locally
LRC

- **ATLAS release -** A version of the Atlas software for data analysis / regular snapshots required to be installed by all (Atlas) sites

Autopilot – Utility to manage pilot job submission to site queues via Condor-G.

- **CE** - consists of one or more similar computers, managed by a single scheduler/job queue, which is set up to accept and run grid jobs. The machines do not need to be identical, but must have the same OS and the same processor architecture. In OSG, the CE runs the bulk of the OSG software stack.
- **Condor-G** – Interfaces: GRAM, ClassAd, PKI. Reference: http://www.cs.wisc.edu/condor/condorg/
- **GLEXEC** - -- Permits a pilot job to re-authenticate using the credential of the payload user, as if the user had directly submitted the job. A program to make the required mapping between the grid world and the Unix notion of users and groups, and has the capacity to enforce that mapping by modifying the uid and gids of running processes. Reference: https://www.nikhef.nl/pub/projects/grid/gridwiki/index.php/GLExec
- **Globus** – the Globus Toolkit http://www.globus.org/
- **GRAM** - service that "provides a single interface for requesting and using remote system resources for the execution of 'jobs'. The most common use of GRAM is remote job submission and control. It is designed to provide a uniform, flexible interface to job scheduling systems." Technical Scope: RPC & Basic HTTP framing, RSL, PKI. References: http://dev.globus.org/wiki/GRAM, http://www.globus.org/alliance/publications/papers.php#Resource%20Management%20Components.

GSIFTP , GridFTP- An implementation of ftp that uses Grid Proxies for authentication and authorization and is compatible with popular tools such as globus-url-copy (from the globus toolkit). Interfaces: FTP, PKI. Reference: http://dev.globus.org/wiki/GridFTP,

LFC – LHC File Catalog. System for authenticated registration of mappings between logical file names and one or more physical file names (i.e. replicas). https://twiki.cern.ch/twiki/bin/view/LCG/LfcAdminGuide

LRMS – Local Resource Management System, i.e. whatever batch system is in use at the site (Condor, SGE, PBS, etc).

OSG Client - The OSG CLI client suite. Includes all end-user tools to handle credentials, look up information, and submit jobs to sites. Software stack distributed by means of VDT

- **Panda Monitor -** The Panda graphic web application interface by which users and administrators can see the state of the system.
- Panda Server -- The Panda job scheduling/dispatch server. This is where pilot jobs retrieve real user payloads.

Pathena - The CLI utility by which ATLAS users can submit jobs that use the ATLAS analysis framework (Athena) to Panda.

- **SRM** - middleware components that manage shared storage resources on the Grid and provide: uniform access to heterogeneous storage; ftp negotiation; dynamic TURL allocation; access to permanent and temporary types of storage; advanced space and file reservation; and, reliable transfer services. Interfaces: WebServices, XML, PKI Reference: https://sdm.lbl.gov/srm-wg/doc/SRM.v2.2.html, http://www.dcache.org/manuals/dcache-workshop-Sep-2005/dcache-workshop-Sep-2005-timur-srm.pdf
- **VOMS** – a system that manages real-time user authorization information for a VO. VOMS is designed to maintain only general information regarding the relationship of the user with his VO, e.g., groups he belongs to, certificate-related information, and capabilities he should present to resource providers for special processing needs. It maintains no personal identifying information besides the certificate. Interfaces: GSI/TCP, SQL, WebServices, PKI. Reference: http://edg-wp2.web.cern.ch/edg-wp2/security/voms/voms.html

1.1. ATLAS Top 5 Issues

In this deployment, ATLAS has identified several issues that need to be addressed by OSG. They are:

1.1.1. OIM/Gratia/WLCG COORDINATION

Specific problem: Namespace issue(s) with probes, OIM. Resource group vs. resource vs. "site" in EGEE.

General Architecture Issue: Coordination between OSG subsystem developers. This is only likely to get worse as OSG becomes more integrated rather than a loose collection of technologies.

1.1.2. WLCG/SAM/RSV/VO TESTS

Specific Problem
For WLCG the agreed upon monitoring solution was for RSV probes to upload results to SAM. This works OK for grid-level testing, but in EGEE, the SAM system is designed to allow VO's to define VO-specific tests which are then centrally submitted (as jobs) and reported on centrally
In OSG, RSV probes are run locally and they report centrally. But there doesn't appear to be a provision for VO-specific probes. Either a VO would need to manually add them per site, or the VO probes would need to be distributed with the OSG stack--neither of which is very scalable.

General Architecture Issue:
RSV makes VO-specific tests difficult because it is distributed rather than central. Some solution or workaround for VO-defined tests is probably needed.

1.1.3. GLOBUS GATEKEEPER

Specific problem: Gatekeepers at BNL have experienced load-related failures. Throttling would prevent this.

General Architecture Issue: OSG CE robustness, scalability, and high availability.

Robustness: The gatekeeper has no self-protection mechanism. It will attempt to service unlimited requests until the system load peaks (~500) and slows to a crawl or hangs (sometimes requiring a reboot).
Scalability: 2000 job limit. Given the lack of protection, we deliberately spread load across multiple gatekeepers.
HA: But because the Globus gatekeeper is a stateful service, it cannot easily be clustered such that multiple instances can sit behind a virtual IP. Which leads to having to hard-code the usage of multiple gatekeepers at a site.

1.1.4. GLEXEC DEPLOYMENT

Specific problem:
gLExec (worker-node based re-authentication) must be installed on top of the OSG worker-node client, but refuses to trust any files residing on a network filesystem (e.g. AFS or NFS). But most sites have historically installed the worker-node client on NFS via pacman for use by a cluster. This situation requires a local install of the osg client, which, at 455MB) is much larger than most cluster admins would prefer.

General Architecture Issue: The use of monolithic pacman installs as a deployment method. Lack of full dependency separation between components that might allow separate partial installations.

1.1.5. FUTURE COMPONENTS (STORAGE, CE, ETC?)

Specific Instance(s):
Addition of Bestman/Xrootd to VDT was very rough and slow. Is the Globus gatekeeper to remain the only CE used in OSG?
Can CREAM be clustered behind a load-balancing/high-availability switch? Mainly this is a question about how server state is stored (database?, files?) and how it deals with server/host certificates.

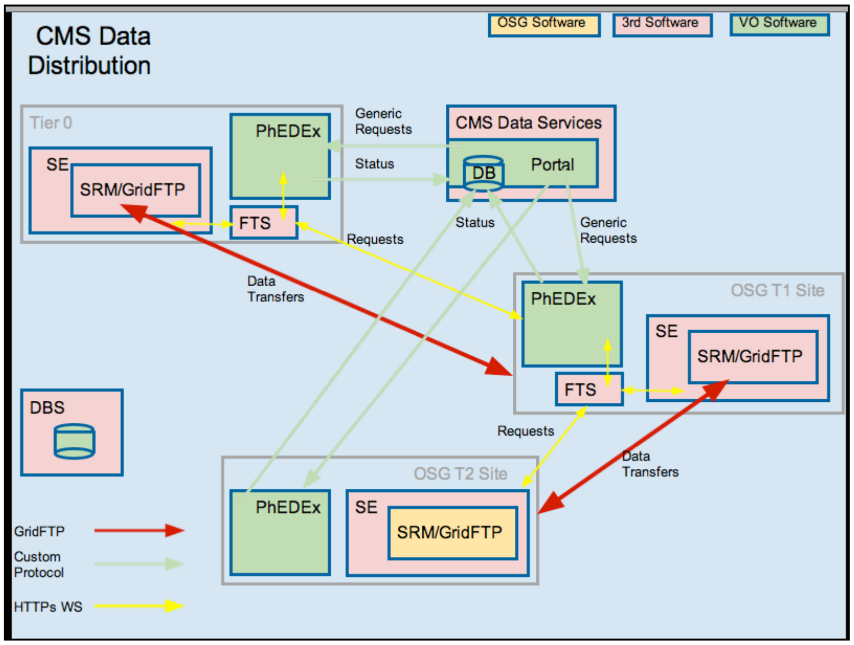
General Architecture Issue:
Whether to provide for a plugin architecture for integration of new technologies rather than approaching each new technology de-novo. This question may apply to other components in addition to Storage (CE, workload management).

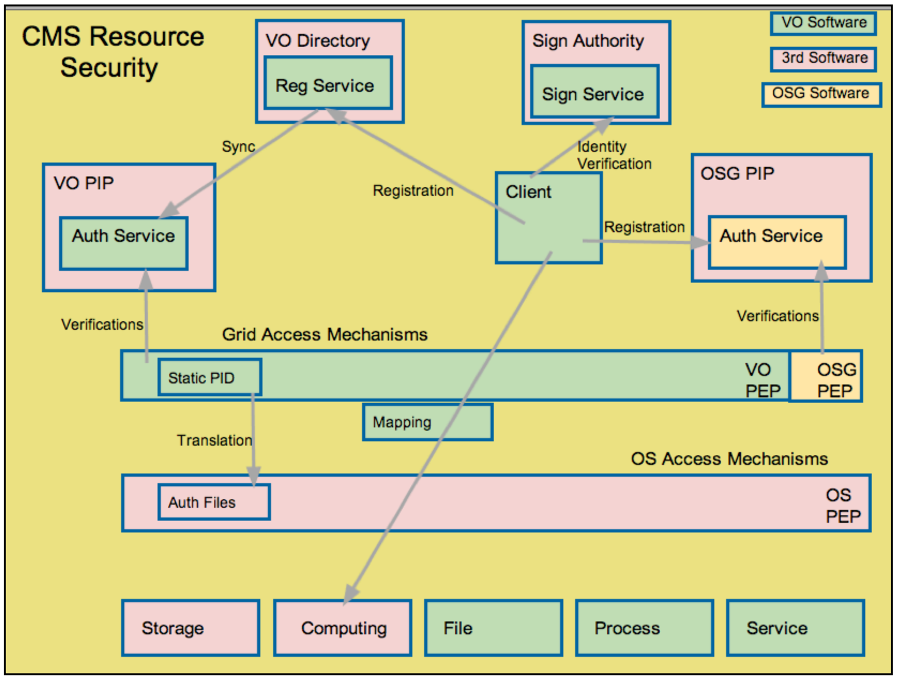
1.1.6. CEMON

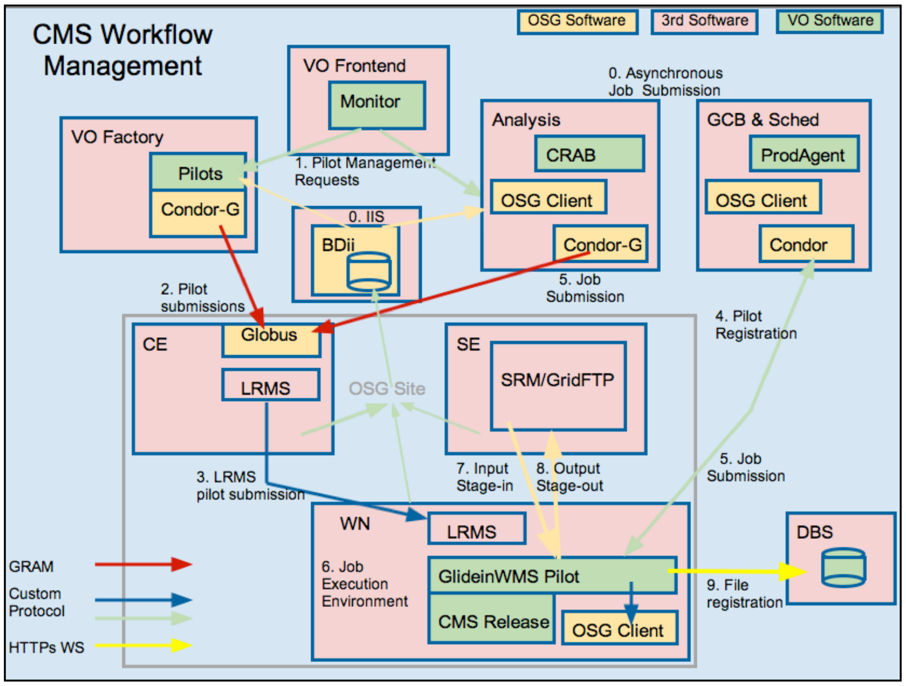
Specific Instance: Complexity and robustness issues of CEMON client as a translator from LDiff to ClassAds.

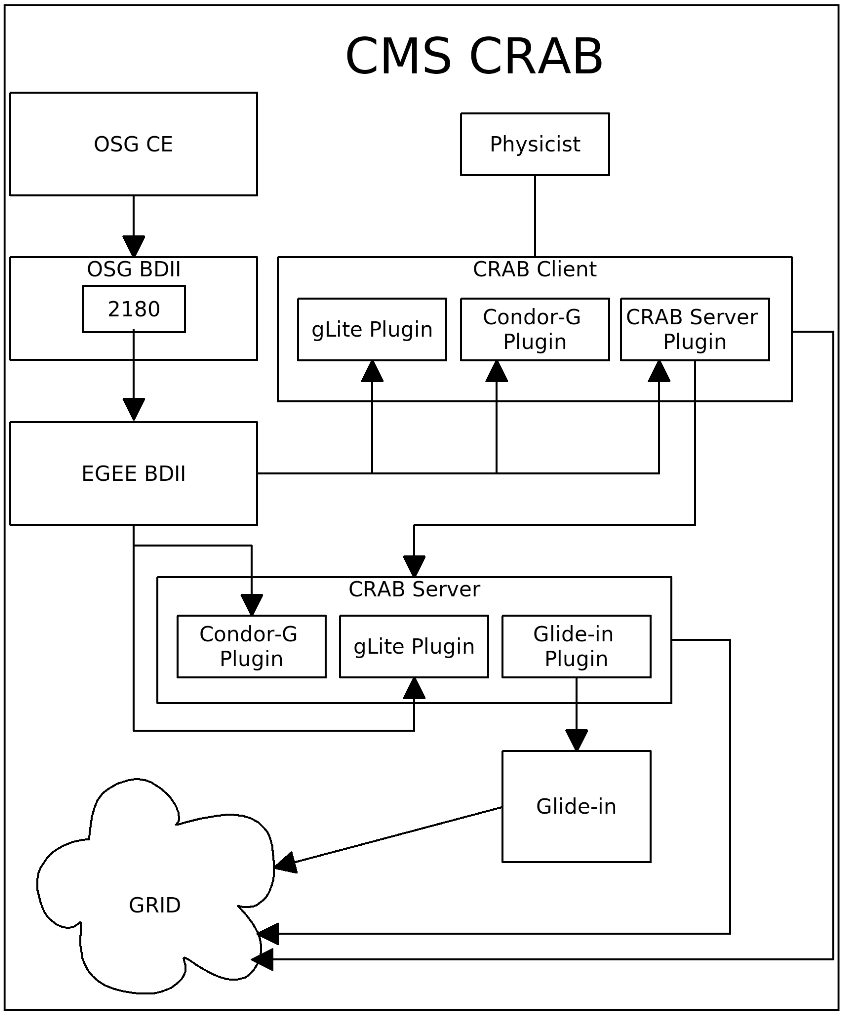
2. CMS

For CMS, the Data Distribution, Workflow Scheduling and Resource Security diagrams are presented next. They incorporate like in the ATLAS case various VO developed and owned components.


CMS Data Distribution
OSG Software
3rd Software
VO Software
Tier 0
PhEDEx
SE
SRM/GridFTP
FTS
Generic Requests
Status
CMS Data Services
DB
Portal
Status
Generic Requests
Requests
Data Transfers
OSG T1 Site
PhEDEx
SE
SRM/GridFTP
FTS
DBS
Requests
Data Transfers
OSG T2 Site
PhEDEx
SE
SRM/GridFTP
GridFTP
Custom Protocol
HTTPs WS


CMS Resource Security
VO Directory
Reg Service
Sign Authority
Sign Service
VO Software
3rd Software
OSG Software
Sync
Identity Verification
VO PIP
Auth Service
Registration
Client
OSG PIP
Registration
Auth Service
Verifications
Verifications
Grid Access Mechanisms
Static PID
VO PEP
OSG PEP
Mapping
Translation
OS Access Mechanisms
Auth Files
OS PEP
Storage
Computing
File
Process
Service


CMS Workflow Management
OSG Software
3rd Software
VO Software
VO Frontend
Monitor
VO Factory
Pilots
Condor-G
1. Pilot Management Requests
0. Asynchronous Job Submission
Analysis
CRAB
OSG Client
Condor-G
GCB & Sched
ProdAgent
OSG Client
Condor
0. IIS
BDii
2. Pilot submissions
5. Job Submission
4. Pilot Registration
CE
Globus
LRMS
OSG Site
SE
SRM/GridFTP
3. LRMS pilot submission
7. Input Stage-in
8. Output Stage-out
5. Job Submission
WN
LRMS
6. Job Execution Environment
GlideinWMS Pilot
CMS Release
OSG Client
DBS
9. File registration
GRAM
Custom Protocol
HTTPs WS



- **CE** - consists of one or more similar computers, managed by a single scheduler/job queue, which is set up to accept and run grid jobs. The machines do not need to be identical, but must have the same OS and the same processor architecture. In OSG, the CE runs the bulk of the OSG software stack.
- **CMS Release** - A version of the CMS software for data analysis / regular snapshots required to be installed by all (CMS) sites
- **Condor-G** – Interfaces: GRAM, ClassAd, PKI. Reference: http://www.cs.wisc.edu/condor/condorg/

DBS – http://cmsdbs.cern.ch/

- **GlideinWMS** – Interfaces: Condor-G, BDII, PKI. Reference: http://www.uscms.org/SoftwareComputing/Grid/WMS/glideinWMS
- **GLOBUS/GRAM** - service that "provides a single interface for requesting and using remote system resources for the execution of 'jobs'. The most common use of GRAM is remote job submission and control. It is designed to provide a uniform, flexible interface to job scheduling systems." Technical Scope: RPC & Basic HTTP framing, RSL, PKI. References: http://dev.globus.org/wiki/GRAM, http://www.globus.org/alliance/publications/papers.php#Resource%20Management%20Components.

- **GSIFTP/GridFTP** - An implementation of ftp that uses Grid Proxies for authentication and authorization and is compatible with popular tools such as globus-url-copy (from the globus toolkit). Interfaces: FTP, PKI. Reference: http://dev.globus.org/wiki/GridFTP,
- **LRMS** - local resource manager system and represents a software component responsible for computing resource management. Well known examples are Condor, PBS, or LSF.
- **Monitor** - a component of the glideinWMS system the VO Frontend monitor keeps track of the amount of requested work vs. available glidein pilots and also sends requests whenever required.
- **OSG Client** - software stack distributed by means of VDT
- **Pacman** middleware component used in OSG for packaging and distribution of software, developed at Boston U. Read more at http://physics.bu.edu/~youssef/pacman/.
- **Pilots** - glidein instances running on sites that register with the ProdAgent and complete work (virtual condor nodes)
- **PKI** - http://www.pki-page.org/

ProdAgent - https://twiki.cern.ch/twiki/bin/viewauth/CMS/ProdAgent

- **SE** The interface through which Grid components communicate with a storage unit that has 1) protocol for local data access protocol; 2) protocol for secure wide-area transfer; 3) exposes status of availability and space; and 4) includes an SRM interface. The SE is a sufficiently flexible interface to grid storage units that allows interoperability of the grid Tier System without forcing local sites to change their existing software stack.
- **SRM** - middleware components that manage shared storage resources on the Grid and provide: uniform access to heterogeneous storage; ftp negotiation; dynamic TURL allocation; access to permanent and temporary types of storage; advanced space and file reservation; and, reliable transfer services. Interfaces: WebServices, XML, PKI Reference: https://sdm.lbl.gov/srm-wg/doc/SRM.v2.2.html, http://www.dcache.org/manuals/dcache-workshop-Sep-2005/dcache-workshop-Sep-2005-timur-srm.pdf
- **VO Factory** - glideinWMS component responsible for providing the right amount of virtual condor nodes for production workloads.
- **VOMS** – a system that manages real-time user authorization information for a VO. VOMS is designed to maintain only general information regarding the relationship of the user with his VO, e.g., groups he belongs to, certificate-related information, and capabilities he should present to resource providers for special processing needs. It maintains no personal identifying information besides the certificate. Interfaces: GSI/TCP, SQL, WebServices, PKI. Reference: http://edg-wp2.web.cern.ch/edg-wp2/security/voms/voms.html

2.1. CMS Top 5 Technical Issues for the OSG

CMS identified issues are as follow.

2.1.1. Batch Scaling Limitations

Condor limitations for over 7000 batch slots and the lack of alternative deployment solutions for large sites – we observed communication problems between components around 7000 batch slots, when the hardware was unable to sustain anymore the required computation and communication requirements

2.1.2. Gatekeeper Overload

CPU overload on the gatekeepers/Condor Scheduler nodes when network connectivity is lost – reconnections take time and also require large amounts of CPU (thousands of jobs)

2.1.3. Authentication & authorization

Authentication and authorization failures when GUMS server is down – we register failures for jobs and we need backup mechanisms like the ones in dCache with priorities for the authorization mechanisms (GUMS, local authorization file - gridmap, etc)

2.1.4. Storage Interoperability

dCache does not interoperate 100% with Hadoop or Bestman. The
Bestman client is preferred due to its low memory footprint. We had to deploy the Bestman client at FNAL to achieve higher transfer rates as well.

2.1.5. CRL management is not complete.

(a) download failures (partial CRL files on disk) fail the entire CRL update mechanism during next iterations and also get the nodes out of service due to missing CRLs ;
(b) out of service nodes (2-3 days) restart with outdated CRLs until the cron job is run (6-12 hours) – this is very bad for jobs
(c) cron job should happen at least two times per day and signal when it fails

2.1.6. Next 4

6. Sometimes site admins need to save their configurations - CMS uses a central CVS for various configurations and this might represent a good element for OSG also
7. For CMS resources it is sometimes difficult to identify errors that occur outside OSG. For example, one WLCG broker had an outdated BDii information and FNAL T1 failed job executions. Other integration projects might face the same issue.
8. Various monitoring systems do not necessarily take in consideration that dCache is a batch-like system. RSV is highly configurable but still has a 10 minutes default timeout – we had to increase this timeout to 1 hour
9. We have encountered several situations where the CRL or certificate download of one CA fails for several days – there is no automated method to take in consideration these transient failures until they are fixed in RSV ; lately the RSV probes for the date on disk instead of the CRL expiration date

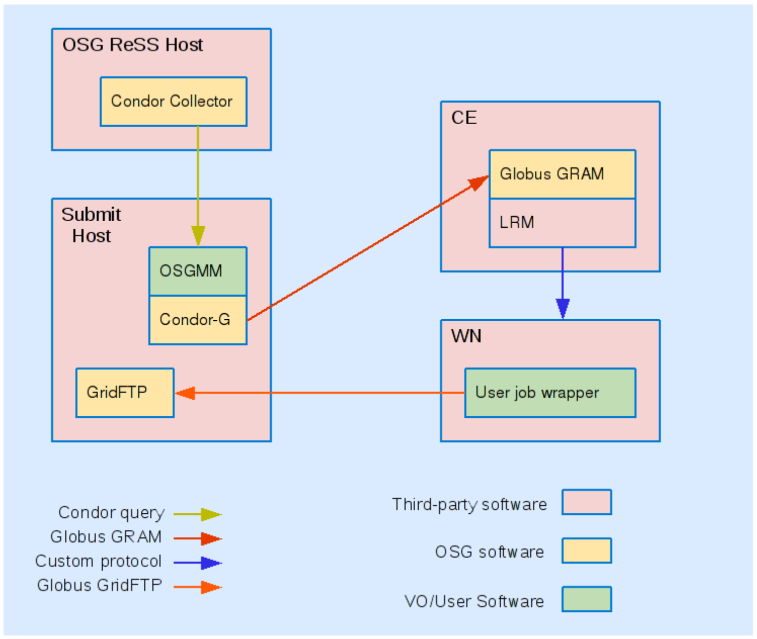
3. LIGO VO

The main benefit that LIGO has drawn from OSG over the years is through the VDT. LIGO generally has been happy and hopes to continue to draw value from this aspect of OSG. To do that, the VO has asked for native packaging for CentOS and Debian (i.e. rpms and debs) of the software. This is extremely important. Over the years, LIGO's members have used lots of different mechanisms to retrieve the software for collaboration. In the end, native (by OS) packaging was the best solution found.

Based on LIGO's opinion, if OSG is to become a service provider to a broader set of communities, OSG needs to supply a base OSG software suite natively packaged and tested for various operating systems. It should be as easy as "yum groupinstall osg" on RedHat based systems, followed by a service starts to be ready to submit jobs onto OSG. (Of course, there will still be the credential and authorization process, but that's another story.)

The OSG architecture also needs to simplify the process of getting jobs onto OSG resources by reducing the site specific knowledge needed by users. There appears to be a large overhead on users to understand the configurations at different OSG sites in order to really be able to run jobs effectively. Watching LIGO users trying to use OSG and similar resources, the most promising approach seen is based on pilot jobs to create a large condor pool using OSG resources. For LIGO, such a system would be invisible to the users. They would log in as usual and submit jobs in much the same way as they do now (admittedly there would be a few changes to how users describe their jobs, but that is needed even internally to LIGO Data Grid at this time). So the question is: who should build such pilot job systems? Is it experiment/user dependent?

4. Engage VO



The Engagement VO relies on a simple OSGMM/Condor-G setup to provide match making between jobs and sites based on job requirements and site status. Site status is determined by OSGMM verification jobs and a moving window of job success/failures. For data management, there are two major use cases in Engagement. The first one is when there is a static dataset which all jobs require. The dataset can be pushed out by OSGMM and its availability advertised so that jobs can match against resources where the data exists. The second use case is when jobs require a job specific data set. In this case, GridFTP is used within the job to transfer inputs/outputs.

4.1. Engagement Top 5 issues:

No job outflow support in OSG software stack.
The OSG software stack currently provides mechanisms for accepting jobs, but does not provide functionality for local jobs to spill onto OSG. This means that it is hard to convince universities and projects with existing infrastructure to adopt OSG. In the case where there is campus infrastructure, the user must explicitly choose upfront whether to send jobs locally, or to OSG.

Information system not monitored well.
Because OSGMM gets its key information from the OSG wide ReSS collector, it is sensitive to CEMon issues at the site level. There is currently no or very little monitoring taking place to alert resource owners and/or OSG staff when a site is misconfigured or not reporting to the

information system. Additionally it is not clear what the correct forum is for discussion new requirements for site publishing. As new requirements are developed, a VO such as Engage may choose to probe systems for certain characteristics, then advertise at the VO level, however in some (possibly many) of these cases, the correct long term solution for OSG is to push this advertisement directly to the source, the resource provider who can authoritatively assert a characteristic of the resource, versus a derived understanding from a probe.

SRM usability.
Engagement has been asked by several OSG groups to use SRM for storage management. However, we believe that the usability is too difficult for Engage end users at this time.

VOMRS/VOMS/Gratia for mixed science VOs
VOMRS/VOMS/Gratia currently does not provide enough functionality/integration for mixed science VO when it comes to the grouping of users and reporting usage based on science. The controlled vocabulary for assigning researchers to science domains appears to need some work as well. We believe that ultimately, the users must self-describe themselves, and be responsible for maintaining their science associations. We must have the tooling and workflow that enables (and requires) them to do so.

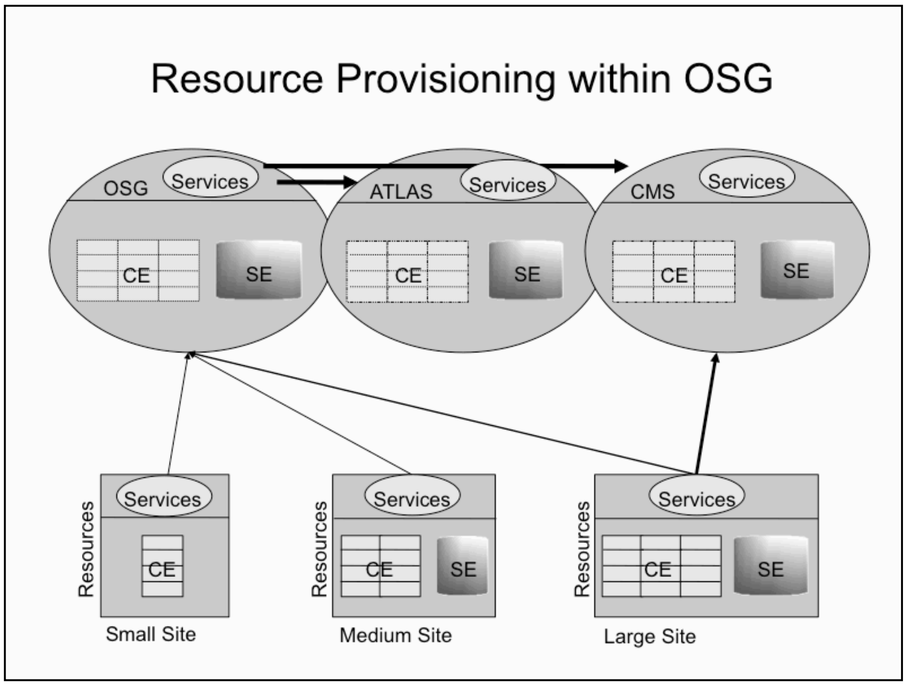
4.2. Infrastructure for Engage Hosted Services

The Engage VO depends heavily on a set of hosted services and infrastructure. These systems are currently hosted at RENCI, and residing on hardware that is sufficient at this time, but nearing end of warranty. McGee has an action item to more fully document these systems, which will be done within a few weeks. If the engage program is to continue and grow, then a more formal operational environment for these systems is necessary and appropriate.

5. Technology Architecture

1. Introduction

The overall architecture of the OSG allows for VO services accessing a set of distributed resources through a common infrastructure. This is depicted in a very simplified and abstracted way in the following picture, where OSG is seen as a VO itself, while the sites are various resource providers, namely universities and laboratories. Resources are supported by various projects and grants, while the allocation policies varies from completely opportunistic to very clearly stated ones. For example, in the US-CMS production environment these resources must be used by the VO members whenever



2. Hardware

For each VO, the hardware architecture to access resources through the OSG consists of *submit and client resources* (computers, disks, databases) where Users submit jobs or data; *Virtual Organization (VO) management resources* where a group of Users manage their VO; *Identity, Compute and/or Storage hardware at a site* which are the computers, disks and tape where the management of the individual identities, the jobs and data are executed and managed remotely.

Any or all of this hardware can be shared by multiple VOs based on the policy of the owner and agreements/delegation between the owner and the VO. The OSG itself accepts agreements to host VO services on OSG hardware, as an example to run the VOMS server on the same hardware as the OSG VO server.

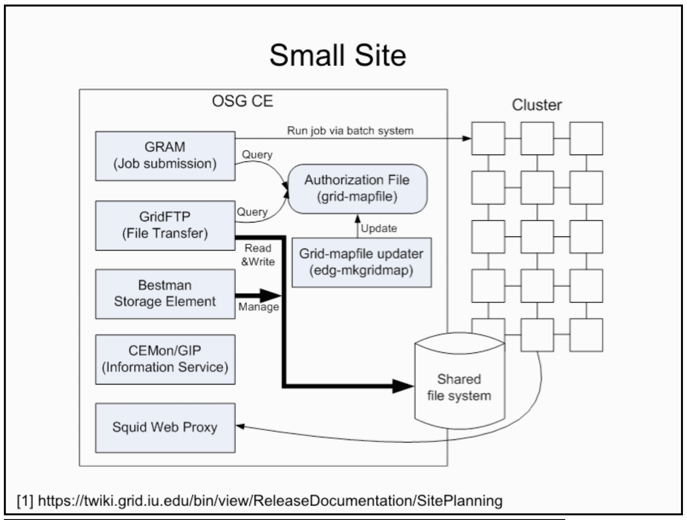
Given the pace of change of hardware it is difficult to provide specific guidance of the hardware to be purchased. The OSG collaborative twiki gives the ongoing /current information:
https://twiki.grid.iu.edu/bin/view/ReleaseDocumentation/SitePlanning
https://twiki.grid.iu.edu/bin/view/ReleaseDocumentation/HardwareRecommendations

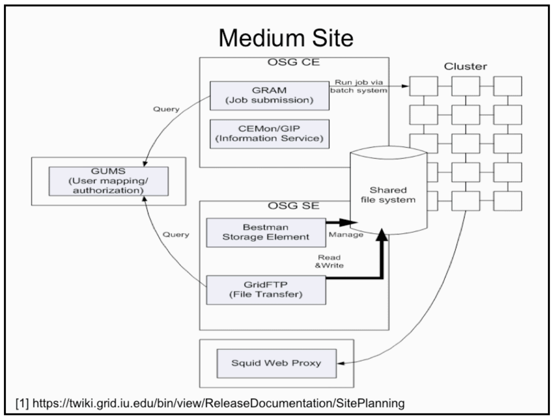
https://twiki.grid.iu.edu/bin/view/Documentation/StorageDcacheOverviewHardwareLayout
https://twiki.grid.iu.edu/bin/view/Education/RecommendedHardware
https://twiki.grid.iu.edu/bin/view/ReleaseDocumentation/Tier3VMHandsOn

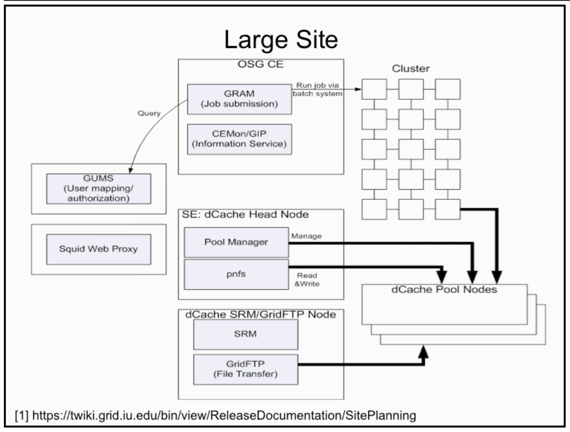
The OSG specific hardware installation needed by:

- An OSG site should allow for minimal additional hardware needed over and above the sources that actually store or transfer data, and those CPUs that execute the jobs.
- A VO should be supportable through hosting of all services by the OSG on behalf of the VO
- A User should allow for job submission and tracking, data movement to and from, monitoring of the progress of the activities and troubleshooting when faults occur, from more than one client system and on any time of submit resource from a laptop or mobile agent to a high end HPC system.
- Addition of a new resource type accessible through the OSG should be non-existent.

An indication of what hardware is needed for different size sites is shown below:



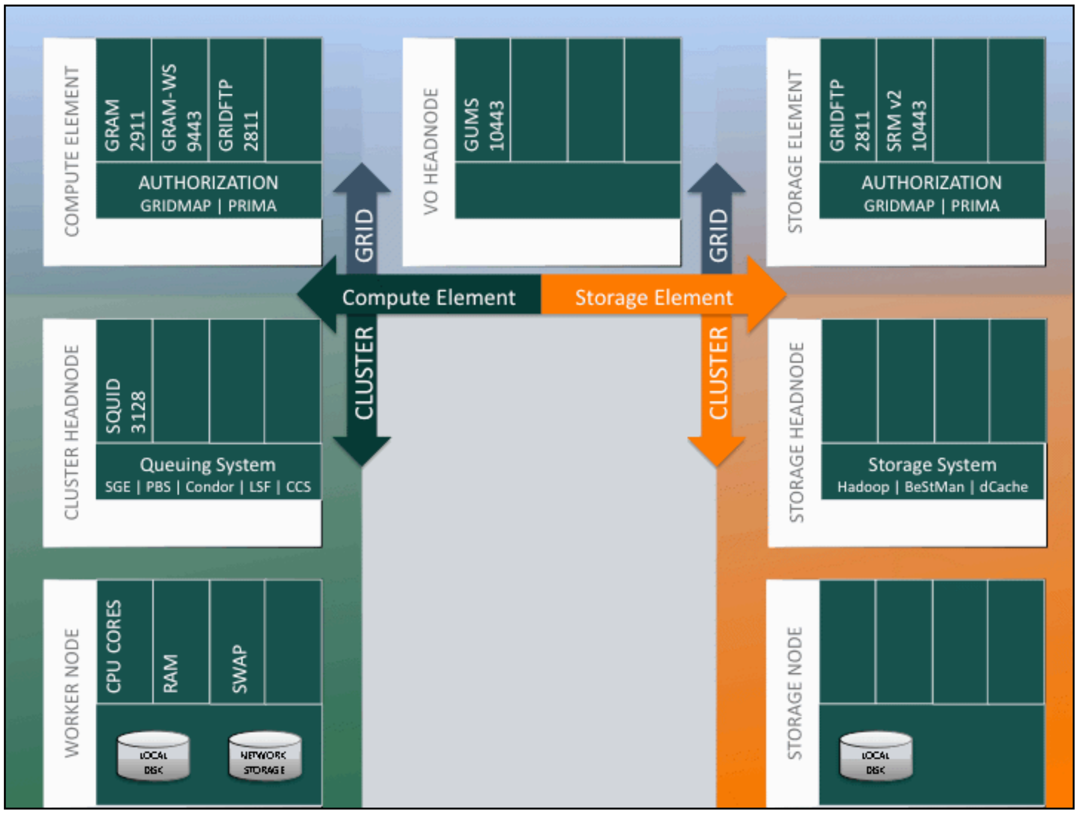


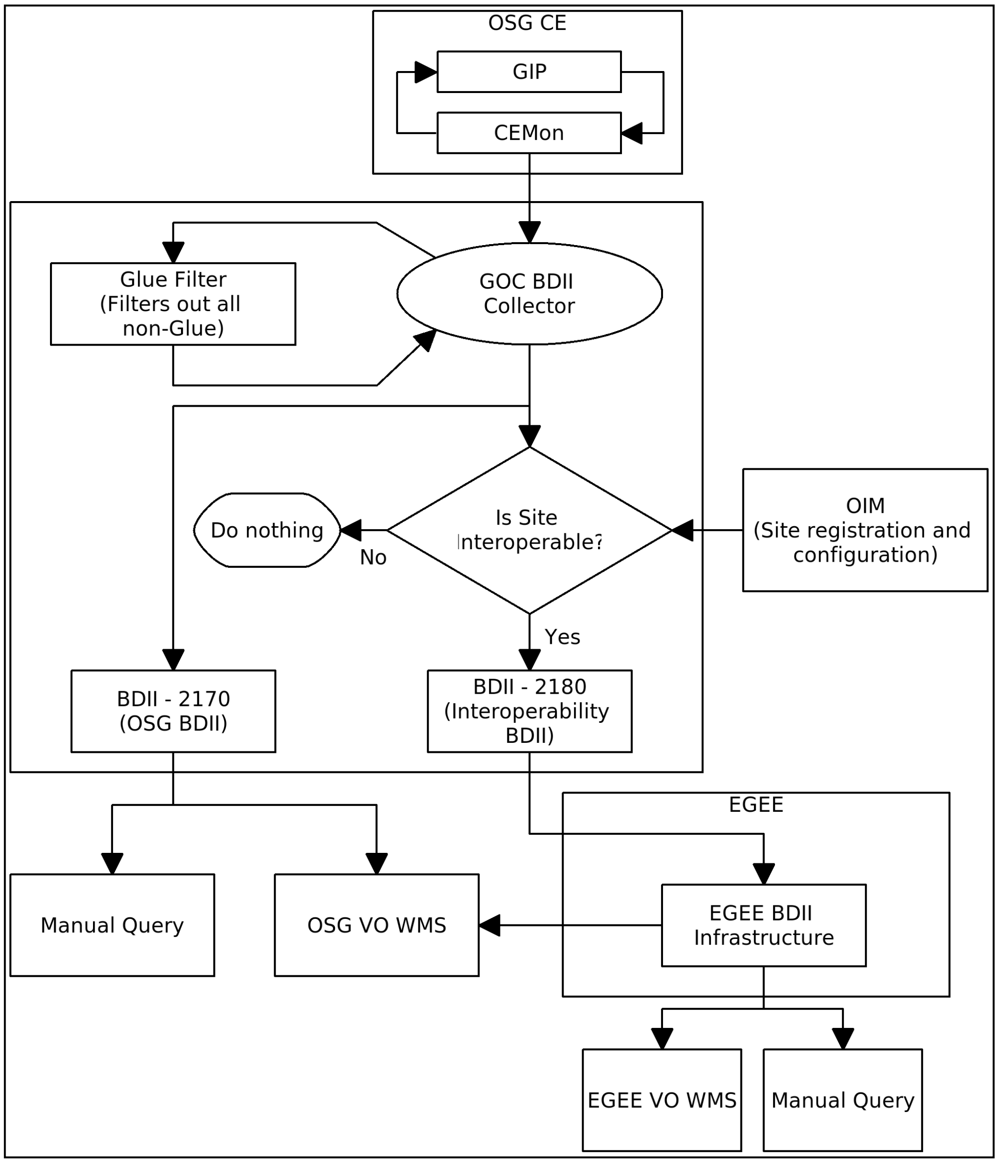


3. Software

The architectural software components that contribute to the OSG are listed below with pointers to the documentation that gives their description and interfaces.

3.1. Basic Services





OSG Information Services

VO Interfaces – Diagrams, more than one copy, interrelationship, - Diagrams and URL. Diagrams of 3 typical sites in the install documentation. Specific instances but don't show you the variety of implementations. Glossary point to the glossary. Protocols. Want the Interfaces. – Nov 13th

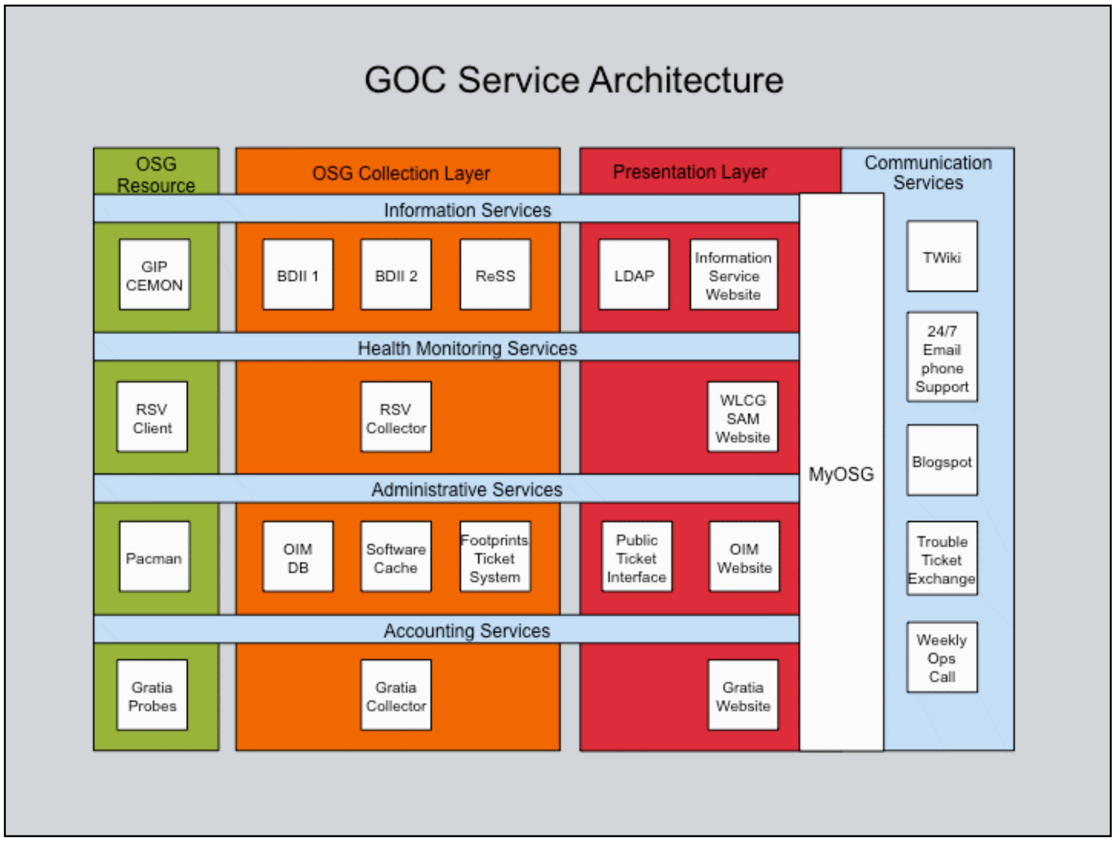
- **Authz** Interfaces: OGSA, XML, PKI. Reference: http://forge.gridforum.org/projects/ogsa-authz
- **BDII** - (Berkeley Database Information Index) is an LCG implementation of Globus GIIS-like information index based on the Berkeley Database. Interfaces: LDAP, TCP, PKI Reference: http://glite.web.cern.ch/glite/packages/R3.1/deployment/glite-BDII/glite-BDII.asp,
- **CE** - consists of one or more similar computers, managed by a single scheduler/job queue, which is set up to accept and run grid jobs. The machines do not need to be identical, but must have the same OS and the same processor architecture. In OSG, the CE runs the bulk of the OSG software stack.

- **Condor-G** – Interfaces: GRAM, ClassAd, PKI. Reference: http://www.cs.wisc.edu/condor/condorg/
- **GIP** - a configurable LDAP information provider that differentiates between static and dynamic information. OSG sites use GIP to advertise a variety of grid-related configuration data. GIP is interoperable with LCG.
- **GlideinWMS** – Interfaces: Condor-G, BDII, PKI. Reference: http://www.uscms.org/SoftwareComputing/Grid/WMS/glideinWMS
- **Glue Schema** - http://forge.ogf.org/sf/projects/glue-wg
- **GRAM** - service that "provides a single interface for requesting and using remote system resources for the execution of 'jobs'. The most common use of GRAM is remote job submission and control. It is designed to provide a uniform, flexible interface to job scheduling systems." Technical Scope: RPC & Basic HTTP framing, RSL, PKI. References: http://dev.globus.org/wiki/GRAM, http://www.globus.org/alliance/publications/papers.php#Resource%20Management%20Components.
- **Gratia Collector** – Interfaces: XML, CSV, SQL - Gratia, tracks VO members' resource usage and presents that information in a consistent Grid-wide view, focusing in particular on CPU and Disk Storage utilization.
- **GSI** - http://www.globus.org/security/
- **GSIFTP** - An implementation of ftp that uses Grid Proxies for authentication and authorization and is compatible with popular tools such as globus-url-copy (from the globus toolkit). Interfaces: FTP, PKI. Reference: http://dev.globus.org/wiki/GridFTP,
- **GUMS** - GUMS is a Grid Identity Mapping Service. It maps the credential for each incoming job at a site to an appropriate site credential, and communicates the mapping to the gatekeeper. GUMS is particularly well suited to a heterogeneous environment with multiple gatekeepers; it allows the implemenation of a single site-wide usage policy, thereby providing better control and security for access to the site's grid resources. See also BNL's site. Interfaces: WebServices, XML, PKI. Reference: https://www.racf.bnl.gov/experiments/usatlas/griddev/gumsdev, https://www.racf.bnl.gov/Facility/GUMS/
- **MyOSG, OIM** – Information and registration system about resources, VOs and services that are registered as part of the OSG. Interfaces: HTTP(s), HTML. Reference: http://myosg.grid.iu.edu/about
- **Pacman** middleware component used in OSG for packaging and distribution of software, developed at Boston U. Read more at http://physics.bu.edu/~youssef/pacman/.
- **PKI** - http://www.pki-page.org/
- **RESS** http://www.cs.wisc.edu/condor/classad Interfaces: WebServices, XML, PKI, Glue Schema, ClassAd (+MatchMaking) Reference: https://twiki.grid.iu.edu/bin/view/ResourceSelection, https://twiki.grid.iu.edu/twiki/pub/ResourceSelection/WebHome/ReSS-ISGC07.pdf
- **RSL** - http://www.globus.org/toolkit/docs/2.4/gram/rsl_spec1.html
- **RSV RSV-SAM** Transport - https://twiki.grid.iu.edu/bin/view/Operations/RsvSAMGridView.
- **SE** The interface through which Grid components communicate with a storage unit that has 1) protocol for local data access protocol; 2) protocol for secure wide-area transfer; 3) exposes status of availability and space; and 4) includes an SRM interface. The SE is a sufficiently flexible interface to grid storage units that allows interoperability of the grid Tier System without forcing local sites to change their existing software stack.
- **SRM** - middleware components that manage shared storage resources on the Grid and provide: uniform access to heterogeneous storage; ftp negotiation; dynamic TURL allocation;

access to permanent and temporary types of storage; advanced space and file reservation; and, reliable transfer services. Interfaces: WebServices, XML, PKI Reference: https://sdm.lbl.gov/srm-wg/doc/SRM.v2.2.html, http://www.dcache.org/manuals/dcache-workshop-Sep-2005/dcache-workshop-Sep-2005-timur-srm.pdf

- **VOMRS** – Interfaces: SQL, WebServices, HTML, HTTP. Reference: https://twiki.grid.iu.edu/bin/view/Integration/VOMRS.
- **VOMS** – a system that manages real-time user authorization information for a VO. VOMS is designed to maintain only general information regarding the relationship of the user with his VO, e.g., groups he belongs to, certificate-related information, and capabilities he should present to resource providers for special processing needs. It maintains no personal identifying information besides the certificate. Interfaces: GSI/TCP, SQL, WebServices, PKI. Reference: http://edg-wp2.web.cern.ch/edg-wp2/security/voms/voms.html

3.2. Operations Services



- **BDII** - (Berkeley Database Information Index) is an LCG implementation of Globus GIIS-like information index based on the Berkeley Database. Interfaces: LDAP, TCP, PKI Reference: http://glite.web.cern.ch/glite/packages/R3.1/deployment/glite-BDII/glite-BDII.asp,
- **Blogspot**
- **CEMON** – translates from LDAP to ClassAds
- **Footprints Ticket Interface**
- **Gratia Collector** – Interfaces: XML, CSV, SQL - Gratia, tracks VO members' resource usage and presents that information in a consistent Grid-wide view, focusing in particular on CPU and Disk Storage utilization.

- **GUMS** - GUMS is a Grid Identity Mapping Service. It maps the credential for each incoming job at a site to an appropriate site credential, and communicates the mapping to the gatekeeper. GUMS is particularly well suited to a heterogeneous environment with multiple gatekeepers; it allows the implemenation of a single site-wide usage policy, thereby providing better control and security for access to the site's grid resources. See also BNL's site. Interfaces: WebServices, XML, PKI. Reference: https://www.racf.bnl.gov/experiments/usatlas/griddev/gumsdev, https://www.racf.bnl.gov/Facility/GUMS/
- **MyOSG, OIM** – Information and registration system about resources, VOs and services that are registered as part of the OSG. Interfaces: HTTP(s), HTML. Reference: http://myosg.grid.iu.edu/about
- **Pacman** middleware component used in OSG for packaging and distribution of software, developed at Boston U. Read more at http://physics.bu.edu/~youssef/pacman/.
- **PKI** - http://www.pki-page.org/
- **RESS** http://www.cs.wisc.edu/condor/classad Interfaces: WebServices, XML, PKI, Glue Schema, ClassAd (+MatchMaking) Reference: https://twiki.grid.iu.edu/bin/view/ResourceSelection, https://twiki.grid.iu.edu/twiki/pub/ResourceSelection/WebHome/ReSS-ISGC07.pdf
- **RSV RSV-SAM** Transport - https://twiki.grid.iu.edu/bin/view/Operations/RsvSAMGridView.
- **Software Cache**
- **VOMS** – a system that manages real-time user authorization information for a VO. VOMS is designed to maintain only general information regarding the relationship of the user with his VO, e.g., groups he belongs to, certificate-related information, and capabilities he should present to resource providers for special processing needs. It maintains no personal identifying information besides the certificate. Interfaces: GSI/TCP, SQL, WebServices, PKI. Reference: http://edg-wp2.web.cern.ch/edg-wp2/security/voms/voms.html.

6. Notes from Technology Meeting

Notes from the Technology Group Meeting

DMWM

- Phedex
- CRAB
- DBS
- Web tools
- Prodagent
- CMS dedicated sites

PANDA

- Local drop box

Cache documents at the Site with ability to query.
No deletion of log files at the site.
Quill? Jobmon cache?
Schedd on the side.
Query local collector

CRAB team /GlideinWMS roll out – Frank, Sanjay, Eric, analysis operations; h/w for glideinwms.

Testing and built in software quality
http: statements, https + smart switches.

Architecture Document audience

- Documentation
- VOs – dependencies
- Managers and evolution
- Developers satellite proposals

OSG delegated services and resources
OSG delegates services and resources to other VOs.

1. CMS Analysis of Several Design Choices in OSG and their Implications

1.1. Information systems

Information systems for OSG are independently developed and there is no effort or rule for interoperability. While there is a push for the storage systems' interoperability, there is none for the monitoring or publishing systems. The interoperability with other Grids and similar infrastructures (like WLCG) is done on a-per-system basis without generic rules. There was considerable effort on the FNAL/CMS part to debug various WLCG monitoring test failures caused several times by how exchanged information was performed. In one case, for example, a WLCG BDii was retrieving and processing inadequately OSG's published data.
The accuracy of these systems is also important for workflow scheduling.

1.2. Scheduling workflows

Scheduling workflows over OSG is a difficult problem yet for the participating VOs. There are ongoing efforts that address the lack of Grid-level scheduling mechanisms but without a large acceptance on the OSG. VOs have created their own mechanisms for tackling the problem (like Pegasus or Condor's Glidein) and if these solutions will become part of OSG software distribution, they can address this missing key software component. VOs will instantiate easily their own scheduling infrastructure. WLCG's resource brokers (RB) and workload management systems (WMS) have not yet a counterpart in the OSG architecture, even though as mentioned some VOs have developed their own solutions.

1.3. Data Handling and Management

Data handling and management solutions for OSG are in their infancy. The Globus level mechanisms are used to move data and users keep track by themselves for the locations of their datasets. By comparison, CMS has implemented and used for more than 4 years now a complete solution for Gridlevel data management – PhEDEx. This package is responsible for tracking data set locations and their movements based on well defined subscription requests.

Similar software solutions need to be identified or devised and incorporated within the VDT in order for the VOs to be able to provide higher level services for data tracking. This can greatly help the VO-level scheduling problem because having computer-aware information about data handling, then the job scheduling mechanisms can implement advanced algorithms for job and workflow placement based on data location as well.

1.4. Debugging

While the mechanisms for sending jobs over the OSG have been addressed with priority, the associated debugging tools have been neglected so far. Users relay in most cases on the basic error codes returned directly from the Grid software or they build their own debugging tools. OSG can pursue this problem and provide to the users specific tools or have site install additional helper ones.

By comparison in CMS users have access to a dashboard system that records certain events about each submitted job and have extensive documentation about what can be the cause for each error. In addition, any job is run through special wrappers that keep track of the local events and provide the entire execution logs to the users once completed. Experts can also provide support when these errors are complex by inspecting the logs and the status of the used resources.

CMS has deployed the CondorView solution for providing extensive cluster status to the users. In addition, recently FNAL CMS T1 has adopted the JobMon tool developed at INFN-Pisa. JobMon is an extended Job Monitoring tool developed to allow users to look at their running jobs, including a job's log and error files, in real time without the need for direct access to the execution WNs. It has helped not only the users but also the site administrators themselves by giving a structured view of the job execution processes and their logs in association with other events on the T1 resources. The interface itself supports advanced queries for different patterns.

1.5. Portals and user interfaces

Portals and user interfaces are missing from the OSG design, while in CMS, for example, they represent a central tool for the community. Users and site executives use various portal tools for the daily tasks from visualization of updated statistics about resources and jobs to projects' status tracking and development.

1.6. CRL Management

The certificate and revocation lists management has undergone major improvements over the last years. Automated mechanisms are in place for their management. However, errors are not yet fully understood or handled accordingly. These errors come inherently with the implemented mechanisms and alternative methods might have provided a better reliability of these processes. The lack of OSG supported CRL caches represents another missing here. Sites are left to deal by themselves with the retrieval of these CRLs. CA caches are already in place but not open to the site usage or advertised yet.

2. General

2.1. Topics to be Communicated

Condor & Dagman tools for workflows
How OSG works and what are its components
How the physics experiments use OSG infrastructure to move and distribute data
How OSG can work with scientists to include workflow capabilities in their proposal

7. Enterprise Architecture Notes

1. Overview

Enterprise architecture is the science of designing an enterprise in order to rationalize its processes and organisation. Most of the time, the highest level of rationalisation lead to automatisation. Enterprise architecture is both the process and the product of this process of rationalisation. As a complex process, Enterprise architecture must repose on a framework, methodologies and tools. A formal definition of the structure of an enterprise comes from the MIT Center for Information Systems Research:

Enterprise architecture is the organizing logic for business processes and IT infrastructure reflecting the integration and standardization requirements of the firm's operating model.

Enterprise architecture describes enterprise applications and systems with their relationships to enterprise business goals. Another comprehensive definition of enterprise architecture is provided by the IFEAD (Institute for Enterprise Architecture Developments) as:
Enterprise architecture is a complete expression of the enterprise; a master plan which "acts as a collaboration force" between aspects of business planning such as goals, visions, strategies and governance principles; aspects of business operations such as business terms, organization structures, processes and data; aspects of automation such as information systems and databases; and the enabling technological infrastructure of the business such as computers, operating systems and networks.

2. Methods and frameworks

Enterprise architects use various business methods and tools to understand and document the structure of an enterprise. In doing so, they produce documents and models, together called artifacts. These artifacts describe the logical organization of business strategies, metrics, business capabilities, business processes, information resources, business systems, and networking infrastructure within the enterprise.
A complete collection of these artifacts, sufficient to describe the enterprise in useful ways, could be considered an'enterprise' level architectural description, or an enterprise architecture, for short. This is the definition of enterprise architecture implied by the popular TOGAF architectural framework.
An enterprise architecture framework is a collection of tools, process models, and guidance used by architects to assist in the production of organization-specific architectural descriptions. See the related article on enterprise architecture frameworks for further information.

3. Areas of practice

Many enterprise architecture frameworks break down the practice of developing artifacts into four practice areas. This allows the enterprise to be described from four important viewpoints. By taking this approach, enterprise architects can assure their business stakeholders that they have provided sufficient information for effective decision making.
These practice areas are
Business:

1. Strategy maps, goals, corporate policies, Operating Model
2. Functional decompositions (e.g. IDEF0, SADT), business capabilities and organizational models expressed as enterprise / line of business business architecture
3. Business processes, Workflow and Rules that articulate the assigned authorities, responsibilities and policies

4. Organization cycles, periods and timing
5. Suppliers of hardware, software, and services

Information:

1. Metadata - data that describes your enterprise data elements
2. Data models: conceptual expressed as enterprise information architectures, logical, and physical

Applications:

1. Application software inventories and diagrams, expressed as conceptual / functional or system enterprise / line of business architectures
2. Interfaces between applications - that is: events, messages and data flows
3. Intranet, Extranet, Internet, eCommerce, EDI links with parties within and outside of the organization

Technology:

1. Hardware, platforms, and hosting: servers, and where they are kept
2. Local and wide area networks, Internet connectivity diagrams
3. Operating System
4. Infrastructure software: Application servers, DBMS
5. Programming Languages, etc. expressed as enterprise / line of business technology architecture.

4. Using an enterprise architecture

The primary purpose of describing the architecture of an enterprise is to improve the effectiveness or efficiency of the business itself. This includes innovations in the structure of an organization, the centralization or federation of business processes, the quality and timeliness of business information, or ensuring that money spent on information technology (IT) can be justified. There are many different ways to use this information to improve the functioning of a business. One method, described in the popular TOGAF architectural framework, is to develop an architectural vision, which is a description of the business that represents a "target" or "future state" goal. Once this vision is well understood, a set of intermediate steps are created that illustrate the process of changing from the present situation to the target. These intermediate steps are called "transitional architectures" by TOGAF. Similar methods have been described in other enterprise architecture frameworks.

5. References

- A Comparison of the Top Four Enterprise-Architecture Methodologies, http://msdn.microsoft.com/en-us/library/bb466232.aspx
- Gartner Identifies Ten Enterprise Architecture Pitfalls http://www.ebizq.net/news/11658.html?rss

Future Technology
Configuration Management and Native Packaging

These two technical design decisions can complement one another, since native packaging systems provide an integrated post-install configuration hook (rpm-post script in RPMs, and a similar mechanism in DPGK).

There are two ways in which the situation with OSG software is different.

First, while most typical Linux software only needs to know about the host upon which it is being installed in order to be configured properly, typical OSG software needs information about the distributed environment in which it is going to work. It needs information about the site layout (storage, Gratia collector, batch system, GUMS host, etc.) and grid-level configuration (OSG Gratia collector, BDII, etc).

Second, in the context of a Linux distribution, the post-install (or post-upgrade) package script is a one-time mechanism. It is only triggered on install--normally there is no ongoing configuration mechanism. Typical Linux distribution vendors assume manual administration after install. But for OSG site administrators, it would be very useful for OSG software to be fully integrated with an ongoing configuration management mechanism, both at install time and afterwards.