EGI-InSPIRE

SCIENCE GATEWAY PRIMER

Document link	https://documents.egi.eu/document/1463
Last modified	29/01/2013
Version	0.9.1
Document Status	Public - draft

Abstract

This document …



I. COPYRIGHT NOTICE

II. **AUTHORS LIST** (alphabetic order)

Name	Affiliation	Role*
Balaz, Antun	Scientific Computing Laboratory, Institute of Physics Belgrade, Pregrevica 118, 11080 Belgrade, Serbia.	NIL
Cauhé, Elisa	Instituto de Biocomputación y Física de Sistemas Complejos, C/ Mariano Esquillor s/n, 50018 Zaragoza, Spain.	SCI-BUS
Chen, Hsin-Yen	Academia Sinica Grid Computing Centre (ASGC), Academia Sinica, 128, Sec. 2, Academia Rd., Nankang, Taipei 11529, Taiwan.	
Jovanovic, Petar	Scientific Computing Laboratory, Institute of Physics Belgrade, Pregrevica 118, 11080 Belgrade, Serbia.	NGI
Kacsuk, Peter	Computer and Automation Research Institute, Hungarian Academy of Sciences (MTA SZTAKI), Kende u. 13-17, Budapest H-1111, Hungary.	SCI-BUS
Loureiro-Ferreira, Nuno	EGI.eu, Science Park 140, 1098 XG Amsterdam, The Netherlands.	VT coordinator, EGI-InSPIRE
Lovas, Robert	Computer and Automation Research Institute, Hungarian Academy of Sciences (MTA SZTAKI), Kende u. 13-17, Budapest H-1111, Hungary.	VT leader, NIL
Olabarriaga, Silvia D.	Department of Epidemiology, Biostatistics and Bioinformatics of the Academic Medical Center of the University of Amsterdam, The Netherlands.	SCI-BUS, COMMIT
Shahand, Shayan	Department of Epidemiology, Biostatistics and Bioinformatics of the Academic Medical Center of the University of Amsterdam, The Netherlands.	COMMIT
Sudholt, Wibke	CloudBroker GmbH, Technoparkstrasse 1, CH-8005 Zurich, Switzerland.	SCI-BUS
Vudragovic, Dusan	Scientific Computing Laboratory, Institute of Physics Belgrade, Pregrevica 118, 11080 Belgrade, Serbia.	NIL deputy

(*) Virtual team (VT) role in the Science Gateway Primer preparation project. Contributor members come from National Grid Initiatives (NGI) and/or projects (SCI-BUS, EGI-InSPIRE and COMMIT); NIL: NGI International Liaison. See acknowledgments section V and chapter 12 'About the authors' for further information.



III. PARTICIPANTS (alphabetic order)

Name	Affiliation*	Country
Ardizzone, Valeria	Istituto Nazionale di Fisica Nucleare (INFN)	Italy
Astsatryan, Hrachya	Institute for Informatics and Automation Problems, National Academy of Sciences of the Republic of Armenia (IIAP, NAS RA) / NIL.	Armenia
Barbera, Roberto	Istituto Nazionale di Fisica Nucleare (INFN), University of Catania	Italy
Bruno, Riccardo	Consorzio COMETA	Italy
Diaz, Ricardo Graciani	University of Barcelona, DIRAC	Spain
Eigelis, Karolis	EGI-InSPIRE	Netherlands
Glatard, Tristan	Centre National de la Recherche Scientifique (CNRS)	France
Giannakopoulou, Kalliopi	Computer Technology Institute (CTI)	Greece
Gordienko, Yuri	SCI-BUS subcontractor	Ukraine
Gottdank, Tibor	Computer and Automation Research Institute, Hungarian Academy of Sciences (MTA SZTAKI)	Hungary
Huang, Vicky	Academia Sinica Grid Computing (ASGC)	Taiwan
Kiss, Tamas	University of Westminster, SCI-BUS User Support WP leader	United Kingdom
La Rocca, Giuseppe	Istituto Nazionale di Fisica Nucleare (INFN)	Italy
Maffioletti, Sergio	University of Zurich (UZH) / NIL	Switzerland
O'Callaghan, David	NIL	Ireland
Pérez, Rubén Vallés	University of Zaragoza, BIFI, SCI-BUS dissemination WP leader	Spain
Rotondo, Riccardo	Istituto Nazionale di Fisica Nucleare (INFN), University of Catania	Italy
Scardaci, Diego	Istituto Nazionale di Fisica Nucleare (INFN)	Italy
Sipos, Gergely	EGI-InSPIRE	Netherlands
Sjaugi, Muhammad F.	SCI-BUS subcontractor	Malaysia
Sterzel, Mariusz	Academic Computer Centre (ACC) CYFRONET / NIL	Poland
Varga, Kitti	Computer and Automation Research Institute, Hungarian Academy of Sciences (MTA SZTAKI)	Hungary
Vasileios, Gkamas	Computer Technology Institute (CTI)	Greece
Winter, Stephen	University of Westminster, SCI-BUS	United Kingdom
Yen, Eric	Academia Sinica Grid Computing (ASGC)	Taiwan

(*) Participant members are associated to National Grid Initiatives (NGI), projects (SCI-BUS, EGI-InSPIRE) and from public institutions; NIL: NGI International Liaison.

IV. DOCUMENT LOG

Issue	Date	Comment	Author
1	04-11-2012	v0.1 EGI-InSPIRE document template	NLF
2	13-11-2012	v0.2 Merge contributions and editions.	TG
3	19-11-2012	v0.3 Merge contributions and editions.	TG
4	22-11-2012	v0.4 Merge contributions and editions.	TG
5	22-11-2012	v0.5 First public draft version released; minor changes to v0.4	NLF
6	06-12-2012	v0.6 Chapters 3 and 10 updated; other minor corrections.	NLF, SS, HY-C
7	11-12-2012	v0.7 Chapters 1-3 updated; several minor corrections.	SDO, NLF
8	17-12-2012	v0.8 Chapters 3, 6-8, 10 updated; other minor corrections.	PK, DV, SS, TG
9	20-12-2012	v0.9 Chapters 1, 6, 10-12 updated; other minor corrections.	GS, SDO, EC, NLF, TG
9.1	28-01-2013	v0.9.1 Chapter 4 updated; other additions, corrections and comments. Final SGP-VT primer release handed over to Gergely/EGI.eu UCST.	WS, SS, NF

V. ACKNOWLEDGMENTS

The work reported in this document is co-funded by a number of national and international projects, particularly:

- EGI-InSPIRE: A project in the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 283481.
- SCI-BUS: A project in the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement no. RI-283481.
- COMMIT P24."e-Biobanking with imaging for healthcare" funded by the Netherlands Organisation for Scientific Research (NWO)
- Others?

We acknowledge

VI. TERMINOLOGY

A complete glossary of the EGI-InSPIRE project is provided at the following page http://www.egi.eu/about/glossary/.

The following table provides the most frequently used terms of the document.

Term	Definition
Application	Some simulator or data analysis program that will be executed on some infrastructure (via the SG)
Enabling technology	
Science Gateway	Community-specific set of tools, applications, and data collections integrated together via a web portal or a desktop application, providing access to resources and services from an e-infrastructure.
Science Gateway framework	
Science Gateway instance	

VII. EXECUTIVE SUMMARY

TABLE OF CONTENTS

1 INTRODUCTION

Author(s): Robert Lovas

The European Grid Infrastructure (EGI) was established in 2010 as a European-wide federation of national computing and storage resources for multiple research communities. During the 2010-2014 period the EGI-InSPIRE project supports the transition of EGI from a project-based system to a sustainable pan-European e-Infrastructure. In 2011 the EGI-InSPIRE project established a Virtual Team (VT) framework to help the European Grid Infrastructure (EGI) collaboration initiate and run short, focussed projects in the area of non-operational activities around EGI's production infrastructure.

The VT framework enables the members of the EGI collaboration – the National Grid Initiatives, various national and European projects, the EGI.eu organisation – to run projects that support new scientific communities become routine users of the European Grid Infrastructure. Responding to the need for more integrated documentations to those who want to develop community-specific, EGI-based portal or desktop environments – EGI science gateways in short – a new VT project started in this framework in May 2012. The project was titled 'Science Gateway Primer' and had three goals:

1. Gather and provide up to date and complete information about EGI science gateways and science gateway enabling technologies in the EGI Application Database.
2. Give recommendations to EGI-INSPIRE on how to improve the data structure of the EGI Application Database and the EGI website to better support science gateway developers.
3. Write a comprehensive document, an 'EGI gateway primer', that collects information about technologies, policies, solutions that exist within the EGI community for gateway developers.

This document, a primer or 'manual' is the most important output of this project and aims to bring together information, knowledge and recommendations from the EGI community for scientific gateway developers, so they can efficiently design, implement and deliver EGI-based, sustainable science gateway services for scientific communities.

The VT project followed an open approach during the development of this document: the EGI community have been invited to join the VT project on email lists, blogs and forums. The content for the primer emerged from teleconferences and face to face discussions that the 36 VT members held between March-December 2012. Draft versions of the document have been circulated for review to the EGI community and beyond during November-December 2012.

The document ... in chapter 2,... chapter 3, chapter 4, ...

The main motivation for the Primer document is to help developers identify the most suitable set of technologies, to collect and apply best practices and solutions for the development of a domain-specific science gateway. For this purpose the Virtual team created and released the first version of a comprehensive document titled 'Science Gateways Primer' that describes the fundamental definitions (Section 2) and functionalities (Section 3). Then based on the expected science gateway qualities (Section 5) the gateway frameworks and instances are systematically compared (Section 6). The Primer document collects and presents information about the available framework technologies (Section 7), as well as about the individual instances (Section 8) that form together the technology base for the recommended steps to be followed by the targeted audience, the science gateway developers (Section 9). Dedicated parts of the Primer document deal with the integration of EGI policies and the wide range of exploitable services operated by EGI (Section 10), and also with the emerging opportunities and issues of Cloud integration (Section 4).

2 DEFINITIONS

Author(s): Peter Kacsuk, Silvia Olabarriaga, Nuno Ferreira

The term Science Gateway (SG) has been coined by the TeraGrid project in the US, then has been picked up and is currently widely used internationally in various e-infrastructure and e-science collaborations. EGI is one of these and defines a Science Gateway as “a community-specific set of tools, applications, and data collections that are integrated together via a web portal or a desktop application, providing access to resources and services from an e-Infrastructure” [XXX] Nowadays also mobile applications can be included here. These gateways can support a variety of capabilities including workflows, virtualisation software and hardware, visualization as well as resource discovery, job execution services, access to data collections, applications, and tools for data analysis. A science gateway enables community members to define and perform custom research scenarios or other types of use cases.

In this document the terms ‘science gateway’ and ‘gateway’ refer to the same, above definition and are abbreviated as SG.

A SG consists of a set of functional components that mediate between the user (front-end) and e-infrastructure services (back-end). The type and range of functions provided by an SG vary a lot, however a SG is typically a complex system built with a (sophisticated and complex) software stack and involves the expertise of people with complementary roles.

Below we introduce some definitions concerning the most relevant aspects involved.

2.1 "Enabling Technology", "SG Framework", and "SG Instance"

An **enabling technology** provides the required software stack to develop SG frameworks and SG instances. Typical examples of such enabling technologies are:

- Web application containers (Tomcat, Glassfish, etc.)
- Portal or web application frameworks (Liferay, Spring, etc.)
- Database Management Systems (MySQL, etc.)
- Workflow management systems (WS-PGRADE, MOTEUR, etc.)

Gateways can be divided into two main categories: SG framework and SG instance. An introduction and a brief overview of the pros and cons of each category follow.

SG frameworks or generic distributed computing infrastructure (DCI) gateway frameworks each provide a specific set of enabling technologies and front-end and back-end services that together build a generic gateway. SG frameworks are not specialized for a certain scientific area and hence scientists from many different areas can use them. NGIs (National Grid Initiatives) are good candidates to set up such gateways to support their very heterogeneous user communities. For example, the UK NGS, Grid Ireland, Malaysian KnowledgeGrid, etc. set up a P-GRADE portal [2.1] for

such purpose. Typical gateways belonging to this category are Genius [2.2], GridPort [2.3], P-GRADE, Vine Toolkit [2.4], WS-PGRADE/gUSE [2.5], etc. The problem with these gateways is that they can expose a large set of low-level services for their users. Thus, in order to exploit their full power, scientists need a relatively long learning period to efficiently use all the available features. The powerful but complex functionalities offered by a generic SG may be mind-boggling for researchers and their commitment of producing knowledge can be jeopardized.

SG instances or application-specific SGs target a well-defined set of scientists typically working on a specific field of science. They provide a simplified user interface that is highly tailored to the needs of the given scientific community. As a result, the scientists do not have to learn too much for using the services provided by the gateway. On the other hand, these services are limited and hence if a scientist needs a more complex service, for example, utilizing a new type of DCI, this cannot be easily created and managed by these gateways. A typical example is the VisIVO Workflow-Oriented Science Gateway for Astrophysical Visualization developed at INAF [2.6] in Italy. In order to create such SG instances there are two options, which will be described in the following.

The first option is to *write the gateway from scratch*. Since the services needed for a particular community are typically limited, and there are good technologies for the construction of web portals, like Liferay, it is relatively easy to develop such SG instances (compared with the development of a SG framework). However, such simplified gateways typically support the usage of only one particular DCI and possibly do not support some advanced features such as workflow execution. Some communities selecting this option may underestimate the required manpower and time to produce a robust gateway that can be provided as a production 24/7 service for the large number of members of the community. Problems that typically arise once the gateway goes into production and becomes successful are scalability (to cope with more users than initially planned) and flexibility (to add new functions requested by the users). Moreover, while building and maintaining such gateways the different communities usually solve again and again the same technical issues independently from each other, which could be avoided by reusing and customising solutions implemented by others.

~~This redundancy of gateway development effort is a huge waste of cost and manpower. Many times they do not reach the required production level by the time their project is over and then all the efforts building the gateway become useless. Sometimes they manage to produce the gateway in time but when the project is over they do not have the required financial support and manpower to maintain the gateway according to the progress of the underlying software stack. This aspect of maintaining gateways and making them sustainable is often underestimated. Related to this issue is scalability both from the functionality and usability point of view. Once a gateway is provided as a production service, the user community would like to have new functionalities, access to additional types of DCIs, and these wishes often require substantial gateway developments. Another serious problems rises when large number of users start to use the gateway and then turns out that it was not properly designed to serve large number of users in a scalable way.~~

The other option is to *customize an existing versatile SG framework* according to the needs of a certain user community. In this case the full power of the underlying portal framework can be exploited, for example by developing comprehensive and sophisticated workflows for the community and hiding these complex workflows behind a simplified application-specific user interface. This is the approach that was followed by the SystemX project in Switzerland where they developed their proteomics science gateway [2.7] based on P-GRADE and recently on WS-PGRADE. A similar

approach was followed in the UK ProSim project where they created a gateway for biologists to model carbohydrate recognition based on the WS-PGRADE/gUSE portal framework [2.8]. The advantage of this approach is that the DCI access services are solved and provided in a robust way by a SG framework and hence the user communities can concentrate on producing their application-specific layers of the science gateway. In this way the redundancy of developing the same DCI access mechanisms by many different communities can be avoided. Due to the same reason the development time of SG instances can be significantly reduced and there is a good chance that within the lifetime of the requiring project the science gateway can be built and provided as a production service. Obviously the cost of producing such a gateway is usually ~~substantially~~ lower than in the case of the first approach. Since the gateway is a customization of an existing robust and scalable SG framework, the produced production SG instance will also be robust and scalable. The sustainability of such an SG instance is more guarantied than in the case of the first method since the large set of user communities involved in the adaptation and maybe further development of the framework represents a strong lobby force to get further funding for maintenance and development.

It is also important that the gateway framework either should be open source and should involve community members in the development and maintenance of the code. When the SG framework is sustainable, the community of the SG instance should then maintain only a narrow set of user-specific services, and the rest is maintained by the SG framework developer community. This approach is followed by the SCI-BUS (SCIence gateway Based User Support, https://www.sci-bus.eu) project that develops the WS-PGRADE/gUSE based SG framework and derives currently 17 different SG instances for various user communities. Another way to reach sustainability for SG instances is to outsource the enabling technology and SG framework parts to professional (commercial or non-profit) service providers to be able to just concentrate on the domain-specific parts. While this is still limited due to the present funding models of the typical scientific communities and the service offers provided (e.g., by EGI and the NGIs), this approach is expected to become more important together with the rise of cloud computing.

2.2 Front-end, Back-end

In both SG frameworks and SG instances we have to distinguish between two main components:

- Front-end
- Back-end

The role of the **front-end** is to provide the necessary user interface. In the case of SG instances the interface is much customized to the particular needs of the science user community. For example, chemists and biologists would like to see visualization tools for molecules, whereas meteorologists need various types of map visualizations. The major focus on SG instances should be to develop this kind of specialized user interfaces to provide the right front-end for the target user community. In the case of a SG framework the interface is typically more generic, providing a user interface for generic features that might be needed for many different user communities and SG instances. For example, user interfaces for certificate management, file and data management, job submission, workflow creation and management, monitoring, etc. These generic parts of the front-end could also be reused from a SG framework for the implementation of customized SG instances.

~~Quality requirements for a front-end:~~

- **User-friendliness:** provides intuitive user interface.
- **Efficiency:** provides small response time even for complex user requests.
- **Scalability:** provides small response time even for large number of simultaneous user requests.
- **Robustness:** keeps working under any circumstances and recovers gracefully from exceptions.
- **Extensibility:** it must be easy to extend with new interfaces and functionalities.

The **back-end** provides the necessary DCI access mechanisms that are needed to realize the typical gateway functionalities like certificate management, file and data management, job submission, workflow management, monitoring, etc. for various DCIs. The back-end is typically generic, i.e., the same back-end can be used by many different SG instances. Therefore the main advantage of developing SG frameworks and deriving the SG instances for them appears in the field of developing the back-ends. If a generic back-end is developed in a robust way by a SG framework, all the SG instances derived from it can take the benefit from its robustness with no or little development effort. A good back-end can support several DCI types (clusters, grids, desktop grids, clouds, etc.), therefore a distinguishing feature of SG frameworks is how many different DCIs they can support.

Quality requirements for a back-end:

- **Efficiency:** provides small response time even for complex submitted jobs or service calls.
- **Scalability:** provides small response time even for very large number (even for millions) of simultaneous submitted jobs or services calls.
- **Robustness:** keeps working under any circumstances and recovers gracefully from exceptions.
- **Flexibility:** ability to manage many different types of DCIs and many concrete instances of DCIs.
- **Extensibility:** it must be easy to extend with the support of new types of DCIs, with new concrete DCIs, and new back-end services.

2.3 People Roles

People involved in the creation, operation and usage of gateways have different roles and a good gateway should provide support for all the roles.

The first category is the **gateway developers**, who develop the gateways. Here we have to distinguish SG framework developers and SG instance developers. The primary goal of **SG framework developers** is to develop the SG framework back-end in a portable way that enables **SG instance developers** to use it without modifications. Their second goal is to develop the generic part of the front-end, and obviously also generate and maintain documentation up-to-date. This is also important for the SG instance developers and some parts even for the SG instance users, too. The main task of the SG instance developers is to customize a SG framework for their user community. It means that the user interface of the SG framework should typically be extended with new application-specific interfaces. However, if the SG instance is developed from scratch the SG instance developers have the same tasks as the SG framework developers and additionally, developing the application-specific interfaces.

Once the SG frameworks or SG instances are developed they should be set up and operated. Here comes the role of **gateway operators**. They should be able to deploy, configure, run and maintain the gateway service running for the user communities. For these purposes, good gateways provide complete and up-to-date documentation, installation and configuration wizards, user management support interface, etc. These can be developed in a generic way within a SG framework and just be used (maybe adapted) by SG instances.

Once the SG frameworks or SG instances are set up and operated the user can come forward and start to use them. We have to distinguish two user categories: **end-users** and **application developers**. In fact, they need different front-ends. The application developers develop DCI applications, for example new workflows, which are used by the end-users. The application developers are typically IT people or scientists (chemists, etc.) with good understanding of the underlying IT technology. They should get relatively detailed information on the underlying DCIs, while this information could partially or completely be hidden for the end-users. Therefore, the SG frameworks are primarily targeted to the application developers and the SG instances are typically designed for the end-users. Of course, this typical usage does not exclude the possibility that some SG framework can be used by end-users and SG instances can provide front-end necessary for DCI application development. However, the good practice is the clear separation of these two concepts.

It should also be pointed out that SG development, operation and usage rely on the work of several external people's roles, such as suppliers. These include in particular:

- *Software providers* of the scientific software applications which are deployed on the DCIs, interfaced in the gateways and upon which workflows are built.
- *DCI providers* who make the necessary back-end compute and storage infrastructure for the gateways available.

These are usually not directly involved into the ecosystem of a particular SG framework or instance, and clear interfaces are highly recommended here. However, despite this, gateway developers, operators, application developers and sometimes even end-users will often need to communicate and collaborate with these external roles for the clarification of documentation, updates, etc.

3 SG FUNCTIONALITIES

Author(s): Peter Kacsuk, Shayan Shahand, Hsin-Yen Chen, Silvia Olabarriaga

An SG provides an interface between a scientist (or community) and the distributed computing infrastructures (DCIs) – see figure 1. Some of these services are specialized to the scientific domain or community served by the SG. Others have more generic nature, and are found in typically any SG, being the focus in this chapter.

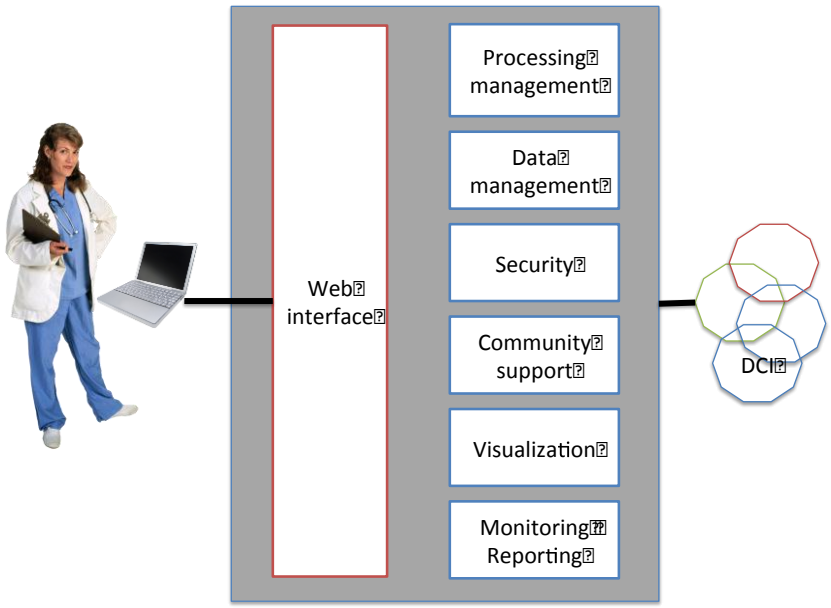

Figure 1 *Functionalities of a SG*

The basic functionalities offered by an SG, collectively via its front-end and back-end, concern in particular:

- **Processing management**
- **Data management**

In addition to these basic functionalities, other important ones are:

- **Security:** provide means to control the access to the gateway and underlying resources;
- **Monitoring and reporting:** provide means to observe the activities/tasks performed via the gateway. This involves both dynamic information, generated while the activity is taking place (monitoring), as well as reporting and statistics about past activity.

- **Community support:** gateways are platforms shared by various users with different expertise that are usually organized and collaborate within research communities. Communication tools are essential to support such communities.
- **Visualization:** since gateways are essentially interactive tools, the visual presentation of data and other information is an important factor.

Each of these topics will be discussed in more detail in the sections below.

3.1 Processing Management

In most of the cases, e-science SGs facilitate access to DCI resources to handle large-scale computations for some scientific applications. Computation is typically performed in "jobs" (batch units of processing) that can be submitted by the SG on behalf of the user. Computation on a DCI can be achieved in various ways.

The following dimensions need to be considered:

- **Type of application**: predefined or user-defined applications.
- **Parallelization**: how tasks at the application level are translated into jobs on the DCI.
- **Workflow execution**: how to execute complex processing pipelines on the DCI(s).
- **Processing on different DCIs**: if the processing should be distributed across DCIs.
- **Scheduling**: how to utilize resources efficiently.
- **Error handling**: what happens when errors occur.
- **Process provenance**: how to record processing activities.

3.1.1 Predefined vs. User-defined Applications

Predefined applications: perform (a set of) fixed processing steps implemented by some well-known software (e.g. AutoDock, BLAST) or a predefined workflow. These applications are stable and optimized for the DCI(s). These applications cannot be changed, which means the users cannot replace their functional components, however their parameters and inputs are configurable by the end-user. A limited set of predefined applications is usually available for DCI users. Based on the authentication mechanism used for the gateway users (see Section 3.3.1) and the policies of DCI providers, users might have only access to predefined applications. This is the typical case of end-users of SG instances.

User-defined applications: if the gateway authentication mechanism and DCI provider policies allow, advanced users should be able to develop and execute their own applications. The gateway frameworks typically support the development cycle, for example by providing means for debugging, tracing, and log inspection. This is the typical case of application developers of SG frameworks.

Whereas predefined applications are easier to use, they are limited to predefined steps. User-defined applications typically require more strict security mechanisms to authenticate the user not only with the SG, but also with the DCI. Furthermore, the trust level has to be high, as all sorts of bad

things may be hidden in user-defined applications as “Trojan horses” on purpose or by mistake (e.g., virus or spam distribution, hacking or DOS attacks, etc. – see 3.3.3).

3.1.2 Parallelization of the application

The first approach that comes to mind is to *submit “jobs” from the application* using the middleware directly. However, this requires modifying the code of the application, being the most invasive and difficult approach. It has the advantage that the application can be efficiently parallelized to reduce latency, and also instrumented to communicate with the gateway for exchanging progress information. This custom approach is only feasible when a small number of well-established applications need to be offered at the gateway, because it requires significant human effort.

The next approach is to write a script (or *wrapper*) that runs the application as a job on the infrastructure without touching the code. The wrapper script takes the executable and input files and submits them to the target DCI with the job description required by the target DCI. The script then downloads the data to the node, runs the application, and uploads the results back to some storage that can be reached by the user. The script can also be instrumented to provide progress information for the gateway. Such a wrapper script is part of the SG enabling technologies, and could be part of an SG framework, since they are typically redundantly developed by each gateway developer community. The workload parallelization can be achieved in a simple way by splitting the data and running the script simultaneously in many resources. The gateway should coordinate data splitting and execution accordingly. The GEMLCA (Grid Execution Management for Legacy Code Architecture) [3.1] developed at the Univ. of Westminster has generalized this script solution and created a generic solution to execute legacy code applications in the grid without requiring the development of individual scripts. This solution is integrated both into the P-GRADE and WS-PGRADE gateway frameworks.

These approaches require custom strategies at the gateway to combine processing pieces that might be needed to accomplish more complex processing pipelines. Another alternative is to use workflow management technology (see below). It should also be noted that a simple splitting of execution pieces is not always possible for complex applications, and works best for so-called “embarrassingly parallel” or “loosely coupled” applications.

3.1.3 Workflow execution

A workflow (WF) is a graph with the nodes that represent jobs or service calls (e.g., Web Service calls) and the directed arcs connecting them represent the data or synchronization dependencies between them. Arcs can be seen as the logical representation of the file transfer mechanism that delivers the necessary input files from a source node to the destination node.

The workflow management system (WfMS) is responsible for executing the nodes of a workflow in the order of the defined dependencies. In a data driven workflow a node can be executed when all of the input arcs of the node have delivered the required input file. It is the task of the WfMS to recognize this situation and then initiate the submission of the job related to the node. The job submission and monitoring is totally controlled by the WfMS.

There are several WfMSs. Some of the most popular ones in Europe are: ASKALON [3.2], Kepler [3.3], MOTEUR [3.4], Taverna [3.5], Triana [3.6], and WS-PGRADE [3.7]. However, many of them are not

integrated with a SG framework. A further problem is that typically they are tightly connected to one particular DCI type (middleware), so if a user community adopts a WfMS, they are consequently bound to a specific DCI type. In order to solve this problem the FP7 SHIWA project created the DCI Bridge [3.8] that enables the access of more than 10 different DCIs including grids, desktop grids, clouds, and local clusters. Since the WS-PGRADE/gUSE WfMS uses the DCI Bridge service it is able to execute workflows on more than 10 different DCI types.

3.1.4 Processing on different DCI types

It is usually the case that SGs provide access to several DCI types, using different types of middleware and operated by different providers with different policies and procedures. In this case a seamless access to computing resources on these DCI types is essential. Different aspects such as authentication, authorization, accounting, scheduling, job/wrapper conversion and coordination between different DCIs should be taken into account to achieve this.

Two solutions can be used for accessing several DCI types from any gateway:

- **SAGA [3.9]:** is an OGF standard solution to access the following DCIs: ARC, gLite, Globus, UNICORE.
- **DCI Bridge [3.8]:** any gateway using the OGF standard BES interface can submit jobs to the following computing infrastructures through the DCI-Bridge: ARC, gLite, GT2, GT4, GT5, UNICORE, BOINC, Amazon, IBM SmartCloud Enterprise, OpenStack, Eucalyptus, OpenNebula (in preparation) [1], LSF, PBS.

3.1.5 Scheduling

The simplest approach to schedule jobs is to use middleware schedulers such as the gLite Workload Management System (WMS) [3.11]. These tools schedule jobs on the DCI resource level leveraging low-level services that operate computing elements such as CREAM service [3.12]. Most WfMS adopt this strategy.

Pilot job frameworks [3.13] are used to provide more flexibility and control at the scheduling time. They send dummy agent jobs (“pilots”) to the middleware schedulers to reserve a resource. Once the resource is available the actual jobs are sent as payloads to those agents. Examples are DIANE [3.14], DIRAC [3.15]. This approach reduces overhead for job submission, allowing for more flexibility and efficiency when dealing with jobs of short or too variable duration.

The highest level of scheduling occurs at the application, where the application developers optimize and control the resource allocation based on the business logic of the application. This requires modification of the application code as well as large integration of the application with the SG.

[1] The clouds are accessed via the CloudBroker Platform [3.10] – a product of CloudBroker GmbH.

3.1.6 Error handling

Grid DCIs provide best effort services and it is not unusual to face situations that eventually cause a job or workflow to fail. Therefore it is important to provide mechanisms to the SG users to deal with failures. The obvious and essential mechanisms are:

- **Pause and resume execution of a job or workflow.** This mechanism is especially useful during scheduled maintenance of the infrastructure, during which the resources are temporarily unavailable but the processing can be resumed afterwards.
- **Retry (restart) from beginning or a check-point of the processing.** This is especially important for long lasting workflows/jobs to avoid restarting them from scratch and losing all the previous computation.

In addition, it is very important to detect, handle, propagate and translate the errors occurred at the various levels of the software (infrastructure, middleware, job, application, workflow, gateway and user). The user and operator should be correctly informed about the errors and the need (or not) of manual intervention.

3.1.7 Provenance

Complex processes are performed on DCI resources. Therefore, it is important to gather process provenance information for later reference, troubleshooting, and audit. Process provenance should be able to answer questions such as:

- Where and when the processing has been done?
- What were the hardware and software specifications of the machine on which the processing has been done?
- Which input data or parameters have been used?
- If there were errors, what type of errors has been detected?
- How much CPU time, memory, and scratch disk have been used?
- What and when was the transition of states during the job runtime (queued, scheduled, running, paused, resumed, failed, finished successfully)?

Such information should be available in a simple way both for the operators of the SG and (advanced) end-users. Capturing provenance is trivial when this is planned from start of the SG development, and can provide a very rich source of information about the SG activity, bottlenecks and possibilities of optimization.

3.2 Data Management

Data is the most important entity in scientific research and therefore a SG is not complete without effective data management facilities. Data operations, user and programmable interfaces, metadata management, access control, and data sharing are some of the main topics in data management that are discussed in this section.

3.2.1 Storage facilities

From the user's point of view the data storages could be categorized into two main groups:

- **Local**: for example local hard drive on the user's desktop, a file server inside the user's institute. Essentially local storages are those that are behind a firewall.
- **Remote**: for example grid storage (SRM), cloud storage, or databases.

3.2.2 Data operations

Files are typically located on different local and remote infrastructures. To use and process these files the following basic data operations should be considered:

- Upload from local storages to remote storage
- Download from remote storage to local storage
- 3rd party file transfers between two remote storage
- Delete and edit operations on remote storage
- Rename files on remote and local storage
- List/select files in the local and remote storage

Different protocols and different credentials (see section 3.3) should be used to perform these operations. Metadata attached to the data should persist and expanded after (most of) these operations.

3.2.3 User interface vs. Application programming interface

The operations on files can be initiated either by the user through a SG front-end or by programs through an application programming interface (API). For example, a workflow engine copies files from a local storage to a remote storage before enactment of workflow jobs using the data API.

3.2.4 Metadata

Metadata is a piece of information that describes a piece of data. A rich set of metadata is essential for data discovery, sharing, and processing. Metadata can be categorized into three groups:

- **Generic metadata:** e.g., data format, data and time of creation or modification.
- **Provenance metadata:** e.g., author, means and steps that created the data.
- **Domain specific metadata:** e.g., part of the body in a medical imaging scan.

Standards and ontologies are available to homogenise and facilitate metadata exchange, however the data modelling is usually specific for each scientific domain and community.

Science gateways should enable their users to discover data by providing tools to query on metadata. SGs should also provide tools to modify and curate these metadata. Metadata, especially data provenance, should be kept up-to-date through automatic collection of provenance information during the lifetime of data.

3.2.5 Access Control and Sharing

Data and metadata operations should be controlled based on the access rights of the users and the groups (or communities) that they belong to. SGs should enable data sharing with other users and groups while receiving credit for this (possibly via data provenance metadata - see 3.2.4). Ideally access control needs to be enforced both on the SG level and on the remote and local storage facilities.

In some scientific domains (e.g. medical research) it is very important to build a reliable access control mechanism that can gain the trust and be accepted by the users. Otherwise, the SG will face adoption resistance. Furthermore, in this and other sectors there are legal requirements that have to be followed regarding date storage and access (see 3.3.4).

3.3 Security

Security has three axes: authentication, authorization and accounting (also known as AAA). Because SGs are located between the user space and DCI space, each one of these axes has two faces, one to the user side and one to the DCI side. In other words security is vertical in the software stack and should be enforced from user to DCI and vice versa through transparent and automatic translation of AAA methods. . For example in Chain project…

3.3.1 Authentication and Authorization

Users should be authenticated before getting access to the SGs and consequently to the resources provided by the DCIs through the SGs. Users can be authenticated by one or a mixture of the following methods:

- **Username and password** are provided to the users after registration to the SG.
- **Federated identities:** for example using the same username and password that they use to access the email account in their organization. This method removes the burden of registration and keeping track of username and passwords from the user shoulders.
- **Grid certificate:** users can authenticate with the SG if they have their Grid certificate installed in their browser, example is the X509 based authentication for EGI Operations Portal or the EGI Accounting Portal.

One of the benefits of using SGs is to provide a federated mechanism to authenticate with different DCIs with different types of authentication mechanisms through the gateway. This is possible if the user already defined and stored the proper authentication attributes in his/her account, or authorized the gateway to authenticate on his/her behalf, or some sort of community authentication is provided by the gateway, for example *robot grid certificate*. A robot grid certificate is used by the SG to authenticate itself and get authorized access to grid resources that are utilized by authenticated users in a controlled environment, for example, only to execute predefined applications. If the user has a personal Grid certificate, it would be used to authenticate and get authorized access to grid resources. In this case, the user usually configures the SG to download a proxy certificate from a MyProxy server on his/her behalf. Then the user-defined applications could be run on the respective DCI.

Authenticated users get access to the resources offered by and through the gateway (e.g. computing resources, data, services, applications). Based on the user authentication method and available authentication methods to different DCIs, the authorization differs. SGs should handle authorizations transparently according to the available authentication and authorization methods and policies on both user and DCI side.

3.3.2 Accounting

All user activities should be recorded for later audit, for example to assess damage if some security breach happens or to prevent future incidents. This is particularly important when a community authentication and authorization method (for example a robot grid proxy) is used by the SG. Accounting is also necessary to apply fair use policy, enforce usage quotas, and billing. Accounting is required by the EGI policies (see Section 10).

3.3.3 Application-level security

As already discussed in 3.1.1, applications in SGs may be predefined or user-defined. The former allow detailed security control by the DCI providers, and thus are much less prone to be used a Trojan horses for all sorts of unwanted or illegal activities. The latter have an inherent danger in this respect if not corresponding security policies and procedures are put into place.

For user-defined applications to be run on third-party infrastructure, there are basically two ways that can be followed:

- Either the user that submits the application is trusted to perform this action.
- Or each individual user-defined application is checked before it can be executed.

Within research communities and grid institutions, usually the first way is followed. This is due to already trust existing, and this approach being much easier and faster. However, as e-science grows and becomes more anonymous (e.g., through robot certificates), the second approach could get more important. This is the way that many application stores (e.g., from Apple) and cloud offers (e.g., in the CloudBroker Platform) work.

3.3.4 Legal requirements

In some fields, for example regarding personal and medical data, or regarding potentials for dual use (i.e., applications both suitable for scientific and for military or terrorist utilization), there are rules set by law that have to be followed regarding security. For example, access to infrastructure has to be excluded from certain countries due to sanctions, data have to be anonymized or access to them restricted to a limited set of certified persons, or storage guaranteed to be in a certain legal region.

These rules are often complex, not suitable for modern international e-science collaboration, and bound to certain countries or the EU. Thus it is difficult for each SG provider to make sure that they are followed, in particular as support by specialized lawyers is usually not readily available. A lot of research is anyhow public and therefore less affected by this. Nevertheless, it is usually best to inform oneself from and follow EGI's, NGI's and local institution's policies and procedures here, and

require corresponding certifications from suppliers and users of the SG framework or instance, if necessary.

3.4 Community Support

SGs support complex interactions among (research) groups with diverse expertise. SGs facilitate knowledge exchange among experts from several scientific domains. For example the following experts are involved in a typical neuroscience research project: physicists, medical doctors, image processing experts, computer scientists, project investigators, etc. People take several roles in one or different projects. Roles define functional responsibilities of their holders in a project, examples are: workflow and application developers, workflow optimizer, data curator, e-infrastructure support.

People who are involved in the scientific projects collaborate with each other by sharing data, methodologies and knowledge (expertise, advice). The scope of the project defines the rules of access to these informational resources. This means resources (e.g., files, applications, workflows) might be shared between people and projects. To support collaboration within and among projects, group managements and access control based on roles are necessary, so are tools such as announcement blogs, wikis, forums, and content management systems for communication among users and also between admins and users. For example, the Liferay portal has such features provided out of the box.

Beyond the usual community tools, scientific communities need various repositories where they can share portlets[2], applications, workflows, cloud images, etc. For sharing portlets we recommend the use of the Liferay portlet repository if applying a Liferay-based SG. EGI provides the AppDB[3] repository for storing metadata on applications running on the EGI infrastructures (see details in section 9.2). The myExperiment[4] community repository stores meta-information of various workflow (WF) applications. Taverna [3.5] applications even can be executed from the repository. The SHIWA repository[5] stores not only meta information on WFs but also their execution-related information. SHIWA also provides the SHIWA gateway that enables the execution of the WFs stored in the SHIWA Repository in various DCIs.

3.5 Monitoring and Reporting

SGs should help their users to monitor their processing and data activities, for example, the progress of a workflow execution or file transfer. Process monitoring depends on the methods that are used to communicate with the DCI, the scheduling method, and whether or not the SG is using a WfMS (see Section 3.1 for more details). Similarly data activity monitoring also depends on the technologies that are used in remote and local resources. Like any other portal functionality, security is also a necessity in monitoring. So only those who have permissions can monitor activities based on their

[2] Portlets are pluggable user interface software components that are managed and displayed in a web portal. (From Wikipedia, the free encyclopedia)

[3] http://appdb.egi.eu/

[4] http://www.myexperiment.org/

[5] http://repo.shiwa-workflow.eu/

role, for example, system admins should be able to monitor everything, while group managers should only be able to monitor activities that are happening within their own group.

The monitoring of SGs can be performed from different perspectives and, as a result, various performance reports could be produced. At the lowest level, resource centre administrators perform the fabric monitoring. In order to ensure proper work of SGs built-in functionalities, the SG administrators perform another level of the monitoring (SG server monitoring). The similar is done by SG application developers, which add an application level monitoring on top of this or use the workflow monitoring features of the gateway. This multi-level monitoring approach brings an overlap in the properties that are being monitored, but such an overlap is also beneficial because it provides independent verification of SG functionalities, as well as those of the monitoring systems.

Automatic system monitoring and notification is also essential for (early) event detections such as attack, finished or failed processes, corrupted or missing data. Respective users should get notifications via a configurable notification mechanism if such events are detected.

Orthogonal to monitoring is reporting that automatically generates reports of activities performed in the SG (system-wide), in a project, or by a particular user. Some examples of such activities are CPU/Disk/Memory usage (usage statistics), errors and types of errors. Reporting is also related to accounting (see Sections 3.3.2 and 10.3).

3.6 Visualization

Visualization could be categorized into generic and domain specific visualizations. Examples of generic visualizations are workflow, statistics, and provenance visualizations. These visualizations are meant to help users quickly examining complex system data and events, for example to pinpoint the cause of an error. For example, WS-PGRADE and GVSS gateways [http://dx.doi.org/10.1007/s10723-010-9159-7] visualize job statistics, and the AMC e-bioinfra gateway [http://dx.doi.org/10.1007/s10723-012-9233-4] provides provenance information visualization (such as lineage graph).

Domain specific visualizations usually concern visualizing input/output data, for example to extract new information or examine its correctness. Some examples of domain specific visualizations follow:

- Jmol[6] is a common molecular viewer that is able to visualize protein and ligand in PDB, MOL and XYZ formats. It also has the 2D information of the ligand database that users can use to query and check out the input parameters. This viewer is integrated into the GVSS portal, which enables the user to visualize the protein-compound complex and extract the interaction energy after the job is successfully executed. It also plots the 2D energy histogram or 3D principal components for analysis.
- We need more examples here

There are two mechanisms to implement visualization: server-side and client-side visualization. In the server-side visualization, a computer program is used on the server to render the visual representation of the data and then this visual representation is sent to the client as an image or movie that could be viewed easily with any regular web browsers. In the client-side visualization, the data is transferred to the client where it is visualized by specific software. This method can provide more freedom to the user to interact with the data, however it is more complex to implement

[6] http://jmol.sourceforge.net/

because of web browsers restrictions, capabilities and compatibilities. Technologies such as Java Web Start, Java applet, WebGL, Adobe Flash and Microsoft Silverlight could be used to implement client side visualization.

Finally, visualization can be performed similar to any other application execution steps in SGs performed on DCI resources, for example within the last node of a workflow. While this is the form that is least interactive, it is often required for applications where visualization is very compute or data-intensive.

4 SCIENCE GATEWAYS AND CLOUDS

Author(s): Wibke Sudholt[7]

4.1 Introduction

Cloud computing is one of the most important technology trends that has aroused in the last few years. As such, it is of high importance both for science gateways as well as for EGI as a whole. The goal of this chapter is thus to explore the connection between science gateways and clouds. In particular, it is of interest in which ways science gateways are related to cloud computing, how they are influenced by cloud technology, and from which cloud concepts they might profit.

In general, one may divide between three principle approaches by which science gateways can interact with cloud computing:

- Science gateways can utilize cloud computing resources for the services they offer
- Science gateways can run in the cloud themselves
- Science gateways can adopt the cloud business model

In this chapter, first a short overview about cloud computing will be given. In the following three sections, the approaches listed above will be described in more detail. The chapter then ends by a summary and conclusions section.

4.2 Cloud Computing

4.2.1 Cloud definition

There are many different ways how to define cloud computing (see, e.g., [4.1]), and up to now, no community agreement onto a single definition could be found. An often-used concept is the cloud definition of the US National Institute of Standards and Technology (NIST) [4.2]:

"Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction."

The NIST definition requires five "essential characteristics" from the cloud model:

- "On-demand self-service"
- "Broad network access"
- "Resource pooling"

[7] This chapter is based in part on the presentation of Wibke Sudholt, "Science Gateways and Clouds", Science Gateways session, EGI Technical Forum, Prague, September 20, 2012 (https://indico.egi.eu/indico/contributionDisplay.py?sessionId=48&contribId=252&confId=1019).

- "Rapid elasticity"
- "Measured service"

This means that cloud computing provides immediate access to computer resources via the inter- or intranet, without much initial investment in time, money or expertise. Cloud resources are shared in a multi-tenant model, and provide nearly unlimited scalability, meaning that the resource size can grow and shrink upon request. Metering and corresponding pay-per-use in small steps, often associated with micropayments, result in capital expenditures (CapEx) being substituted by operational expenditures (OpEx). It is important to check each cloud offering one encounters for the principles given here, to make sure that it can really be considered being cloud.

Overall, cloud computing represents not only a change in technology, but more importantly a change in the operational and business model how computer resources are exposed to users. In particular, in cloud computing finally the interface between providers and users is set at the right place, in that the users can chose between the different levels and ways of how the cloud services are offered to and used by him/her. The further sections of this document will explain how this relates to science gateways.

4.2.2 Cloud services

Citing again from the NIST definition of cloud computing [4.2], it separates between three different "service models" how cloud computing is delivered:

- "Software as a Service (SaaS)"
- "Platform as a Service (PaaS)"
- "Infrastructure as a Service (IaaS)"

SaaS means delivery of cloud services on the level of applications. Typically, this means business, office, etc., applications, as exemplified by Salesforce (http://www.salesforce.com) or Google Apps (http://www.google.com/apps/). In the context of science gateways, of course scientific and technical applications are of highest relevance here.

PaaS is usually associated with various programming and deployment frameworks offered in the cloud. Often also additional functionalities such as distribution, messaging, monitoring, databases, etc., are provided. Typical examples include Google App Engine (http://cloud.google.com/appengine/) and core parts of Windows Azure (http://www.windowsazure.com). Of course also science gateways can utilize the corresponding platform tools.

IaaS usually consists of hardware and operating system building blocks such as virtual machines, storage, network, etc., provided in the cloud. Well-known providers here are for example Amazon Web Service (http://aws.amazon.com), Elastic Compute Cloud (EC2) and Simple Storage Service (S3), as well as Rackspace (http://www.rackspace.com). Since science gateways naturally need underlying computer infrastructure, it is obvious that science gateways may also consume such services out of the cloud.

4.2.3 Computing and Processing Considerations

The main consideration regarding where a science gateway should gain its internal or external computing or processing power out of the cloud from is on which level, IaaS, PaaS or SaaS, it is obtained.

IaaS usually provides compute power in form of virtual machines (e.g., Amazon EC2 or OpenStack Nova), the number of which can be dynamically scaled up or down. Here many aspects need to be taken into account, such as the different machine sizes in terms of CPU cores, speed, memory and disk and the flexibility of their definition or selection, the provided network setup regarding speed (e.g., Infiniband) and security (e.g., firewall, VPN), the locations of the data centres, the used virtualization techniques, virtual image generation and sharing, the available operating systems, etc. Generally, a small loss of performance has to be expected when using virtual machines instead of physical machines.

Pricing is mainly measured per hour of running a particular machine instance size in a cloud.

On the PaaS level, there is usually less control about the details of the compute power provided by the cloud. Typically here are scaling factors, which define how many threads or processes may be executed by the selected cloud setup in parallel.

On the SaaS level, computing and processing power is usually directly connected to the provided application, but some flexibility comparable to the IaaS and PaaS levels might be retained.

4.2.4 Data Management Considerations

Clouds can also provide scalable storage for science gateways. Scalable is meant here in terms of size, number of elements, requests, etc.

Looking at cloud storage in relation to data management, one has to be aware of certain characteristics of it, mainly storage access patterns. These patterns can be random access, sequential read/write, low latency, or high throughput depending on the requirements of the application.

The two predominant technological models for cloud storage on the IaaS level are block storage (e.g., Amazon EBS and OpenStack Volume service) and object storage (e.g., Amazon S3 and OpenStack Swift). Whereas block storage most often does not require any changes to the application, as it mimics a disk in a real machine, it may not be scalable in all desired dimensions. On the other hand, it provides random access to files, which is not exactly the case for object storage. Block storage must be treated like a device in a physical machine, and typically requires that it is attached to a running machines instance for access.

Object storage, on the other hand, provides the illusion of an inexhaustible amount of storage space that one can read from or write to via standard interfaces. Object storage, unlike block storage, which can only be attached to one instance at a time, can serve data to or via multiple instances without the need to be replicated. Furthermore, it can be directly accessed from UIs and APIs without being attached to a running machine instance.

For both types of storage, pricing usually reflects GB stored per month and GB transferred into or out of the cloud plus access measures, which block storage normally only offered in fixed sizes.

On the PaaS level, storage is provided in more complex forms that are directly programmatically accessible. In particular, often SQL or non-SQL databases are provided in the cloud.

On the SaaS level, data management is typically automatically included with the application, and not meant to be directly accessed.

Generally, it always has to be taken into account that data management in the cloud will involve data transfer via the internet to or from a cloud provider, including all corresponding speed and security concerns. Some cloud providers also accept the physical sending of storage devices, which might be faster for large amounts of data.

4.2.5 Types of clouds

The final definition that can be found in the NIST document [4.2] is that of four cloud "deployment models":

- "Private cloud"
- "Community cloud"
- "Public cloud"
- "Hybrid cloud"

Private clouds can be in-house or hosted for a particular organization, just for its internal use. They are usually multi-tenant across organization sites, departments, groups and/or users. Their main focus is typically on self-service and accountability inside the organization.

Community clouds go a bit further, in that those clouds are not only for a single organization, but for a bigger, though still separated community across different organizations or individuals. They still follow similar principles, though with a higher emphasis on the sharing aspects of cloud.

Public clouds are offered by certain organizations or cloud providers to a larger community or basically to everybody. This means that they are multi-tenant across organizations and/or individuals. Here the focus is usually on the on-demand and pay-per-use advantages of cloud computing.

Hybrid Clouds finally are a mixture of public, community and/or private clouds. Their focus is typically on providing scalability and failover features such as spill over and cloud bursting.

Science gateways may be related to and/or utilize all forms of clouds, as will be further discussed in the following.

4.2.6 Cloud in EGI

Recent years have seen a huge hype and an exploding market in cloud computing offers, which can thus not be fully presented here. However, the raising interest in clouds has also induced EGI to invest into this technology. Its main goal is "to facilitate the setup of a pan-European federated cloud based on the resources of the NGIs" (http://www.egi.eu/infrastructure/cloud/). For this, a Federated Cloud Task Force has been established (http://www.egi.eu/infrastructure/cloud/cloudtaskforce.html, https://wiki.egi.eu/wiki/Fedcloud-tf). It generates a blueprint document and test bed as well as communicates with providers, users and other influencers.

The EGI Federated Cloud test bed (https://wiki.egi.eu/wiki/Fedcloud-tf:Testbed, https://wiki.egi.eu/wiki/Fedcloud-tf:UserCommunities) consists of resources from various resource providers. It is mainly based on OpenStack (http://www.openstack.org) and OpenNebula (http://opennebula.org) open source IaaS tools. The protocols exposed by the EGI Federated Cloud to users are the Open Cloud Computing Interface (OCCI), Cloud Data Management Interface (CDMI), the Glue schema, and X509 for the authentication (https://wiki.egi.eu/wiki/Fedcloud-tf:UserCommunities#Interfaces_and_protocols).

Beyond EGI itself, also various NGIs, organizations and projects related to it have built up clouds, cloud test beds and cloud projects. The SCI-BUS project, for example, utilizes the CloudBroker Platform (http://www.cloudbroker.com) for its connection to public and private clouds that are in part provided by the project partners.

4.2.7 Further information

Within this Science Gateway Primer, only a very short introduction into cloud computing can be given. A number of documents deal in much more detail with the relevance of cloud computing for science, research and academia. The reader is in particular referred to reports from UC Berkeley [4.3], CERN [4.4] as well as the US Department of Energy [4.5]. Also the European Cloud Computing Strategy (https://ec.europa.eu/digital-agenda/en/european-cloud-computing-strategy) of the European Commission is of importance here.

4.3 Utilization of Clouds

4.3.1 Gateway architecture

In this and the following two sections, we will look in more detail into the three different ways that science gateways can connect to cloud computing, as they were mentioned in the introduction of this chapter.

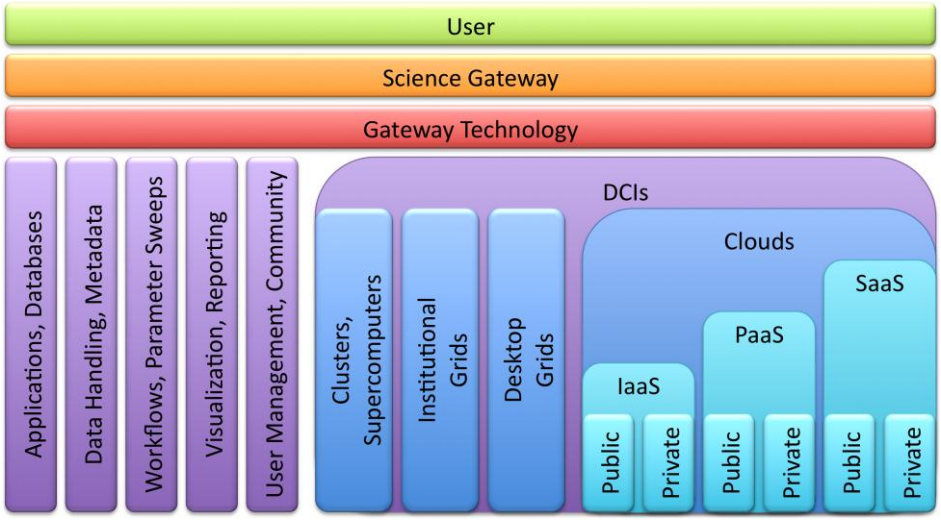

Figure 2 *Architecture for science gateways using cloud services as back-end*

The first and most classical variant is science gateways using cloud services as their back-end. This means that instead of or in addition to clusters, supercomputers and/or grids, clouds are employed as DCIs. Cloud computing thereby often provides a perfect fit for the science gateways, as the back-end DCIs are usually only needed on demand. The figure 2 gives a graphical visualization of the corresponding overall scientific gateway technology stack.

4.3.2 Utilization considerations

As can be seen from the figure above, science gateways can use all sorts of cloud services as DCI:

- **IaaS**, in particular compute and storage resources for scientific calculations and data
- **PaaS**, that is programming and deployment environments for scientific software
- **SaaS**, providing access to complete scientific applications

The biggest difference here is that the lower the level of cloud services that is used, the more functionality (e.g., automation) has to be provided by the science gateway technology itself, and/or the more raw cloud access has to be exposed to the users on the front-end side. On the other hand, using a lower-level cloud service may provide more flexibility in the way it is implemented in the science gateway or how it is utilized by the users. Furthermore, there are not many cloud standards yet, and the functionality of different cloud offers even on the same level might be quite different and incompatible. In any case, this leads to considerable differences in the efforts for building science gateways and/or in the usability that can be provided, as visualized in the Figure above, and thus should be considered in the planning of a science gateway based on clouds. However, it can be expected that more and more generic-purpose technologies that science gateways are constructed upon will include build-in access to various cloud services in the future.

Another important aspect that should be considered when constructing a science gateway on top of cloud DCIs is if private, community, public or hybrid cloud infrastructures should be used. Here among others the following considerations are usually of importance for the gateway developers and operators:

- Which cloud services are already provided by or used by the community that I address with the gateway?
- What are my and/or the considerations of my users regarding security, performance, scalability, variability, failover, geographic location, applicable law, SLAs, certification, vendor lock-in, etc.?
- Can there be a direct payment for the cloud services either by my organization or by the users of my gateway?

Overall, detailed research and planning regarding which cloud level, type and service to use and how is strongly advised.

4.3.3 Examples

There are some science gateways and general gateway technologies that already use cloud services as a back-end. Only a few examples are mentioned here. These might provide guidelines to science gateway users and developers within EGI and elsewhere how clouds can be utilized within their gateways:

- As already mentioned in the cloud computing section, the SCI-BUS project (http://www.sci-bus.eu) employs the CloudBroker Platform (http://www.cloudbroker.com) for its connection to various IaaS clouds. The platform's SaaS and PaaS features are thereby directly built into the WS-PGRADE/gUSE (http://www.guse.hu) framework that SCI-BUS uses as generic-purpose gateway technology, via the CloudBroker Platform APIs.
- Galaxy is an open, web-based platform for data-intensive biomedical research (http://www.galaxyproject.org). It can be used either via a free public server or an own, private installation. Galaxy employs Amazon Web Services (http://aws.amazon.com) to launch Galaxy clusters in the cloud.
- iPlant (http://www.iplantcollaborative.org) targets computational support for plant biology. Its Atmosphere (http://www.iplantcollaborative.org/discover/atmosphere) tool provides a cloud infrastructure service platform that for example allows starting preconfigured virtual machines in the cloud.

4.4 Running in the Cloud

4.4.1 Principles

The second possibility of how science gateways can profit from the advantages of cloud computing is to run the gateway's front-end using cloud services. Again, here all sorts of cloud services can be employed:

- IaaS, PaaS or SaaS
- Private, community, public or hybrid clouds

Also, similar thoughts as in the section about using clouds for the science gateway's back-end above should be applied here in selecting a suitable cloud service for the gateway's back-end:

- How much effort, flexibility and/or usability are suitable?
- What are the considerations for locating and selecting a cloud service?

Of course, both cloud front-end and cloud back-end services can be combined in a single science gateway.

4.4.2 Cloud web support

In many cases, science gateways are nothing more than a special type of website or other online-accessible software with advanced functionality and a distinct audience. Many cloud providers offer a lot of different services for exactly this very generic purpose, which is not specific to science gateways. These can now be used for developing, deploying and running the gateway's front-end.

Typical cloud services for websites employed for science gateways include:

- Utilizing virtual machine instances and block or object storage in public or private IaaS clouds to operate the science gateway web server, database server, etc. and to store the gateway's contents and data.
- Using public or private PaaS environments to develop, deploy and operate the science gateway.

- Creating special sections, installations or derivatives in the public or private SaaS offers of providers of generic-purpose technology for science gateways, or starting from even more generic online website building frameworks.

Combining this with the options to use private, community, public or hybrid cloud setups shows the large number of selection possibilities. Again, detailed research and planning is required here.

4.4.3 Cost considerations

When operating science gateway front-ends within clouds, it should be considered that they usually need to be online all the time, as users expect this. This often makes no difference in private or community clouds, but public clouds generally charge for the usage of time and/or resources. Thus beyond comparing different cloud services, a price comparison with simple hosting offers or own operation should be considered.

However, more points than just the plain direct costs should be taken into account. It may also be useful to utilize cloud resources to prevent any large initial investments, for scalability during peak demands, as well as for backup and failover scenarios. Furthermore, outsourcing to clouds will allow the gateway providers to focus on their core competencies, and thus may free up personnel resources. A full cost of ownership analysis may thus be desirable, although often difficult to perform.

In general, clouds represent a move to more centralization of the IT similar as in the age of mainframe computers. This however means that large opportunities for automation and optimization can be realized, resulting in economy of scale effects. Furthermore, big cloud providers can due to their efficient operation and possibilities such as locating their data centres in regions with direct energy and/or cooling access also often offer “greener” computing and storage than individual local providers.

4.5 Cloud Business Model

4.5.1 Gateway sustainability

The third way how science gateways can take over ideas from cloud computing, deals with business aspects. As already explained in the cloud computing section, cloud computing is more a change in the business model than it is in the underlying technology.

One request that is usually made to science gateways is that over time they need to become sustainable. Of course it does not make sense to offer a gateway if there is not enough user interest in it. It can be observed that there are more and more moves towards metering and billing also academic services. And many governments do not have enough money for large capital expenditures (CapEx) anymore and thus prefer stepwise operational expenditures (OpEx). Unfortunately, though, sponsors are hard to attract to science gateways, and not always gateways may be outsourced or a spin-off company generated from them.

It can thus be expected that for academic and community gateways, base government and NGO funding to build, operate and support the gateways is still needed. However, beyond that one may apply the cloud business model also to science gateways.

4.5.2 Gateway business models

The idea here is to understand science gateways as high-level cloud services themselves, that is as "Gateway as a Service" (GaaS). Then also all corresponding business features from clouds such as pay-per-use and OpEx instead of CapEx can be applied, together with known concepts from web services, social media and open source commercialization.

Some possibilities for setting this up are the following:

- Build science gateways from cloud building blocks
- Charge for not freely accessible resource consumption and application usage
- Have a freemium model with extra charges for additional services
- Offer user subscriptions with different features and prices
- Let commercial and other external users pay for gateway services
- Have a shareholder charge-back model
- Provide and charge for professional consulting, training and support
- Include advertisements and/or user/usage analytics

In the end, each gateway, depending on its features (product) and audience (market), needs to develop its own business model. Of course this requires deep thinking and calculations, and only some initial suggestions can be listed here. Furthermore, the selected business model then also has to be realized on the technical side. However, for example within the SCI-BUS project (http://www.sci-bus.eu), such possibilities are already being explored.

4.6 Summary and Conclusions

In this chapter, the relation of science gateways with cloud computing was explored. After a short overview of clouds in general, three principle ways of how science gateways can profit from the cloud concept were discussed:

- Clouds for gateway back-ends
- Clouds for gateway front-ends
- Cloud as gateway business model

Overall, cloud computing provides a large variety of great opportunities for science gateways. However, detailed considerations are necessary for each individual case. In the future, cloud functionalities included in the base gateway technologies will considerable ease this.

5 SCIENCE GATEWAY QUALITIES

Author(s): Peter Kacsuk

Science gateway (SG) qualities can be accessed from the point of view of different people using the gateway:

- SG developers
- SG operators
- Applications developers
- End-users

5.1 SG Developers

Science gateway developers are mainly concerned with how easy it is to further develop the gateway:

- Open source code or cloud service
- Quality of the code or service including documentation
- Usage of standard software development technologies
- Support for code developers

First of all, the code could be open source for several reasons:

- Avoid vendor lock-in
- Enable the modification or extension according to the specific needs of a certain community
- Developing code by community effort improves the quality of the code (for example, forces the developers to provide better comments, better documentation, etc.).
- Community effort enables more people to work on code development and hence results in shorter time to produce new required functionalities

However, also the inherent risks of basing the science gateway development on a particular open source code should be taken into account. In particular, when the project behind the code in not large enough, not well organized, or not continuously supported by major funding agencies or companies, the support for the open source code and with it its quality, documentation and further development might be insufficient, run out or split. This could lead to a dead end for the science gateway development.

Therefore, intense research should be performed before deciding for using a particular science gateway software base. In particular, in the age of cloud computing, it could become more attractive in the future to use a professional cloud service, which could also be offered by a non-profit organization such as EGI, instead of joining another open source community. Among others, cloud services offer the following advantages:

- Main parts of the service that are not of direct interest for the science gateway developer are developed and maintained by the provider of the service Thus there is much less to learn and worry about for the developer.
- Clean interfaces are provided, which allow the science gateway developer to focus on what is of direct concern for the particular scientific community and to concentrate on developing against these interfaces, instead of having to take the whole software into account. Usually, these interfaces will be open.
- If using such a platform or software as a service solution, it will already provide the necessary scalable hardware alongside with it. This later saves efforts in particular on the deployment and maintenance side of science gateways.

The quality of the code or service can be assessed according to the following criteria:

- Is the gateway software architecture clear, SOA based and/or layered?
- Do the communicating components use existing standards or widely distributed de-facto standards? If there is no standard, then is the developed protocol clear and simple?
- Are the comments sufficient and clear?
- Is the documentation detailed enough and clear?

There are many software development technologies. It is important to follow a state-of-the-art technology that enables the fast and reliable development and release creation. Is there any testing methodology? If yes, the quality of the testing procedure is also a major quality factor. The release policy also should be well-defined.

In order to enlarge the developer community, good support for code developers is important. This can include:

- Forum discussions
- Training materials
- Training courses if required
- Consulting
- Professional services

Quality requirements for the front-end:

- It should be extensible, i.e., it must be easy to extend with new portlets and functionalities.

Quality requirements for the back-end:

- It should be extensible, i.e., it must be easy to extend with the support of new types of DCIs and with new concrete DCIs.

All these aspects are very important for SG framework developers. For SG instance developers some of the requirements can be omitted. For example, to serve a small user community the SG instance code could be proprietary code. In such case the community might be satisfied with less detailed documentation and does not need forum discussions. On the other hand for SG instance developers who would like to derive their SG instance from an SG framework, an additional quality criterion is how this customization procedure is supported in the selected SG framework:

- Is there any customization support?
- If yes, how easy to use this support?

If designed, implemented and operated in a suitable way, cloud services could be used by both SG framework and SG instance developers, albeit probably utilizing interfaces on slightly different levels.

It is the task of the community to choose which of the quality requirements written above will be used for their particular SG instance. The SCI-BUS project would like to apply all the above mentioned criteria both for the SG framework and for the SG instances. SCI-BUS puts significant efforts to develop an easy-to-use customization method in order to derive SG instances from the SCI-BUS SG framework.

5.2 SG Operators

For the science gateway operators the most important quality requirements are:

- How easy to install the gateway? Is there any installation wizard?
- How easy to configure the gateway? Is there any configuration wizard?
- How easy to re-start the gateway?
- How easy to diagnose if a problem occurs? What support the operator can get from the gateway developers?
- How frequent are the gateway releases?
- How easy to update the gateway for new releases?
- How user management is supported? Is there any tool, user interface for this purpose?

Cloud services are of particular interest for SG operation, which to large parts could be taken over by the corresponding service provider.

5.3 Application Developers and End-Users

For the application developers and end-users basically the same quality requirements can be applied and these should separately be investigated for the front-end and back-end.

Quality requirements for the front-end:

- It should provide rich functionalities supporting application development
    - Workflow editor (textual and/or graphical)
    - User-oriented certificate management, or a different solution to authentication and authorization that simplifies and/or automates access for users compared to certificate management
    - Job and workflow submission to various DCIs
    - Execution monitoring to observe and if possible control the execution in the different DCIs
    - Etc. See further functional requirements in chapter 3
- It should be user-friendly and intuitive.
- It should provide user interface (UI) access to an application repository where partially developed and completed applications can be stored and shared with other application developers.
- It should provide UI access to an application repository where ready-to-use applications can be uploaded, stored and provided for end-users.
- It should provide a possibility to deploy and manage applications on the different DCIs.

- It should be efficient, i.e., it should provide small response time even for complex user requests.
- It should be scalable, i.e., it should provide small response time even for large number of simultaneous user requests.
- It should be robust, i.e., it must not be collapsed or frozen under any circumstances.
- It should be extensible, i.e., it must be easy to extend with new portlets and functionalities.

Quality requirements for the back-end:

- It should be flexible, i.e., it must be able to manage many different types of DCIs and many concrete DCIs.
- It should provide monitoring and status information on job/workflow execution.
- It should provide fault-tolerance features that hide the unreliability of the underlying DCI.
- It should be efficient, i.e., it should provide small response time even for complex submitted jobs or services calls.
- It should be scalable, i.e., it should provide small response time even for very large number (even for millions) of simultaneous submitted jobs or services calls.
- It should be robust, i.e., it must not be collapsed or frozen under any circumstances
- It should provide access to an application repository where partially developed and completed applications can be stored and shared with other application developers.
- It should provide access to an application repository where ready-to-use applications can be uploaded, stored and provided for end-users.
- It should provide a possibility to deploy and manage applications on the different DCIs.

6 SCIENCE GATEWAYS LIST AND COMPARISON

Author(s): Elisa Cauhé

In this section, science gateways users can find relevant information about the different frameworks and instances of gateways. Facing the impossibility of listing all the existing science gateways, this document only collects a representative group of them. The following tables show the features they offer according to the chapter 3 “SG Functionalities”.

Note that the last table shows prototypes of science gateways instances (not in production).

6.1 Production Gateway Frameworks

Gateway frameworks	Job management	Workflow editor	Workflow management	Error handling	Supported DCIs	Cloud access	Repository access support	Open source	Monitoring and Reporting	Community Support	Security Authentication Authorization Accounting		
WS-PGRADE/ gUSE kacsuk@sztaki.mta.hu	Yes	Graphical	Yes (own language)	Yes	ARC, gLite, GT2, GT4, GT5, BOINC, PBS, LSF	Via CloudBroker Platform to Amazon, IBM, OpenStack, Eucalyptus	Yes (own repository, SHIWA repository)	Yes (Source-Forge)	Yes	Possible based on Liferay portlets/ technology	Certificate	Virtual organization	No
Genius roberto.barbera@ct.infn.it													
Vine Toolkit krzysztof.													

kurowski@man.poznan.pl													
CT-Science Gateway	Yes	No	No	Yes	Any (JSAGA)	Any (JSAGA)		Yes (Source-Forge)					
GridPort													

6.2 Production Gateway Instances

Gateway instances	Customized from	Job management	Workflow editor	Workflow management	Error handling	Supported DCIs	Cloud access	Repository access support	Open source	Monitoring and Reporting	Community Support	Security Authentication Authorization Accounting		
SZTAKI Autodock Gateway (Docking) kacsuk@sztaki.mta.hu	**WS-PGRADE/gUSE**	Yes	No	Yes (hidden)	Yes	BOINC: EDGeS@ home	No	Yes (own repository)	No	Yes	Yes	Certificate	No	No
RenderFarm (Rendering) julius.tuomisto@laurea.fi	**From scratch**	Yes	No	No	Yes	BOINC	Soon	No	Yes	Yes	Yes	Yes	Yes	No
NRG-CING														
SystemX (Proteomics) peter.kunszt@systemsx.ch	**WS-PGRADE/gUSE**	Yes	Super user	Yes	Yes	ARC, local cluster, clouds via CloudBroker	Yes	Yes	Yes	Yes	Yes but only systemsx: sybit project	Yes, currently only local LDAP, shibboleth coming	Yes - Liferay built in	No
ProSim (Biology) T.Kiss@westminster.ac.uk	**WS-PGRADE/gUSE**													
VisIVO (Astrophisics) ugo.becciani@oact.inaf.it	**WS-PGRADE/gUSE**	Yes	Yes	Yes	Yes	HPC Cluster, gLite	No	Yes	Yes	Yes	Yes	Yes (Liferay)	Yes (Liferay)	Yes (gLite certificate)

MoSGRID (Molecular Simulation) krueger@informatik.uni-tuebingen.de	**WS-PGRADE/gUSE**	Yes	Yes, for experts	Yes, for experts	Yes	D-grid	No	Yes	Soon	Yes	Yes	Certificate	Virtual organization	No
AMC e-Bioinfra Gateway (S) http://www.ebioscience.amc.nl/ebioinfragateway/	**Based on Spring Framework**	Yes	No	Yes	Yes	gLite	No	Yes (only repository)	Soon	Yes	Yes (only experiment support)	Username and Password	Robot Proxy	Yes
AMC e-Bioinfra Gateway (L) http://www.ebioscience.amc.nl/wspgrade34	**WS-PGRADE/gUSE**	Yes	Yes (experts only)	Yes	Yes	gLite, Private Cluster	No	Yes	Yes	Yes	Yes	Username and Password	Personal Certificate and Private key	Yes
EUMED		Yes	No	No	Yes	ANY (JSAGA)	No	Yes	Yes					
CHAIN		Yes	No	No	Yes	ANY (JSAGA)	Yes	Yes	Yes					
GISELA		Yes	No	No	Yes	ANY (JSAGA)	No	Yes	Yes					
DECIDE		Yes	No	No	Yes	ANY (JSAGA)	No	Yes	No					
EARTHSERVER		Yes	No	No	Yes	ANY (JSAGA)	No	Yes	Yes					
agInfra		Yes	No	No	Yes	ANY (JSAGA)	Yes	Yes	Yes					
IGI		Yes	No	No	Yes	ANY (JSAGA)	No	Yes	Yes					

6.3 Prototype Gateway Instances

Gateway instances	Customized from	Job management	Workflow editor	Workflow management	Error handling	Supported DCIs	Cloud access	Repository access support	Open source	Monitoring and Reporting	Community Support	Security Authentication Authorization Accounting		
SZTAKI agInfra Gateway	**WS-PGRADE/gUSE**	Yes	Yes	Yes (hidden)	Yes	gLite, BOINC	No	Yes (own repository)	No	Yes	Yes	X509 cert.	VO based	No
Klios		Yes	No	No	Yes	ANY (JSAGA)	No	Yes	Yes					
COGITO-Med		Yes	No	No	Yes	ANY (JSAGA)	Yes	Yes	Yes					

7 GATEWAY FRAMEWORK DEVELOPERS AND OPERATORS

Author(s): Peter Kacsuk

7.1 What is their Role?

The role of science gateway (SG) framework developers is to develop a robust, scalable and efficient SG framework that can be easily adapted and customized by SG instance developers. SG operators should install gateways and run them as production services for application developers as users.

7.2 What is Expected from Them?

SG developers should be IT people who understand:

- the enabling technologies in order to develop good (user-friendly, robust, scalable and efficient) front-ends,
- the organization and interface of various DCI middlewares in order to develop good (robust, scalable, efficient and flexible) back-ends.

SG framework operators should be IT people who are able to:

- install and configure a SG framework,
- manage users,
- take care of security issues (for example, manage firewalls, etc.),
- regularly update the gateway and the underlying software stack.

7.3 Who are these Players in Europe?

There are several SG framework developer communities in Europe. It is not the goal of this document to show all of them. Rather we show some success stories of gateway frameworks that already work at production level and serve relatively large communities. The other European gateway framework developer communities can be found in the EGI Application Database [7.1].

7.3.1 CloudBroker Platform

The CloudBroker Platform is an easy-to-use application store for high performance computing (HPC) in the cloud. It is developed by the company CloudBroker GmbH, a spin-off of the ETH Zurich in Switzerland, and available in a public version under https://platform.cloudbroker.com as well as as hosted or private versions.

Simply through the user's web browser or through CloudBroker's APIs, users can select from different scientific and technical applications and immediately execute the calculations on several cloud infrastructures, pay-per-use. Users may also register and use their own cloud resources and application software in the platform, profiting from a freemium model in the public version. In the

platform marketplace, users can offer resources and software to other users, and earn money with them being used.

The main characteristics of the CloudBroker Platform are:

- On demand, pay-per-use, license or freemium-based, browser, programmatic and command-line access, thus usable as SG front-end and back-end
- Uses IaaS from cloud providers, offers PaaS for software vendors and provides SaaS for end users
- Management of users, application software, cloud resources, jobs and invoices
- One-stop-shop and marketplace, users can offer own software and resources and use software and resources from others
- Focus on compute-intensive, batch-oriented Linux applications, across different domains
- Adapters to different cloud infrastructures and tools, including Amazon EC2 and S3, IBM SmartCloud Enterprise, OpenStack, Eucalyptus and OpenNebula (in preparation)
- All advantages of a cloud service, such as ease of use, shorter time to market, no need for own high performance computing infrastructure, and operational expenditures instead of capital expenditures

More information about the CloudBroker Platform can be found under http://www.cloudbroker.com. Video demos on the gateway can be found at http://www.sci-bus.eu/demos12.

7.3.2 VineToolkit SG framework

The Vine Toolkit framework is described in [2.4] in detail. Here we cite the abstract of that paper:

“The Vine Toolkit framework integrated with Adobe Flex/BlazeDs technologies. It offers a set of unified and abstract APIs for different grid middleware and rich graphic presentation layer.

Additionally, it automates the integration process with portal frameworks, such as Liferay or GridSphere. Vine Toolkit introduces a concept of subprojects, which extends core APIs or defines new low level components and web applications. This way Science Gateway prototyping process is definitely shortened. Consequently it allows programmers to build software components that can be reused in a simple manner.”

7.3.3 WS-PGRADE/gUSE SG framework

The WS-PGRADE/gUSE generic DCI gateway framework has been developed by MTA SZTAKI in Budapest, Hungary, to support large variety of user communities. It provides a generic purpose, workflow-oriented graphical user interface to create and run workflows on various DCIs including clusters, grids, desktop grids and clouds. The framework can be used by NGIs to support various small user communities who cannot afford to develop their own customized science gateway. The WS-PGRADE/gUSE framework also provides two API interfaces (Application Specific Module API and Remote API) to quickly and easily create application-specific science gateways according to the needs of different user communities. The gUSE back-end supports simultaneous access to several DCI types from the same workflow. In this way not only intra-DCI but even multi-DCI parallelism can be achieved by WS-PGRADE/gUSE and by the gateway instances that were created by its customization. Currently the following DCI types can be accessed from WS-PGRADE/gUSE:



- Clusters: PBS, LSF
- Grids: ARC, gLite, GT2, GT4, GT5, UNICORE
- Desktop grids: BOINC
- Clouds via the CloudBroker Platform: Amazon, IBM, Eucalyptus, OpenStack
- Supercomputers via ARC and UNICORE middleware

More information about WS-PGRADE/gUSE can be found under http://www.guse.hu. Video demos on various features of the gateway can be found at http://www.sci-bus.eu/programme.

8 GATEWAY INSTANCE DEVELOPERS AND OPERATORS

Author(s): Peter Kacsuk and the SCI-BUS community

8.1 What is their Role?

The role of science gateway (SG) instance developers is to develop robust, scalable and efficient SG instances that can be easily used by the target user community and provide them specialized interfaces (e.g. visualization). The target users can be both application developers and end-user scientists.

SG operators should install gateway instances and run them as production services for the user community.

8.2 What is Expected from Them?

SG instance developers should be IT people who understand well:

- the enabling technologies in order to develop good (user-friendly, robust, scalable and efficient) front-ends,
- the organization and interface of various DCI middleware in order
    - to develop good (robust, scalable, efficient and flexible) back-ends if the gateway instance is developed from scratch or the gateway framework selected as the basis of the gateway instance should be extended with access to new DCIs.
    - to be able to install and run the gateway framework selected as the basis of the gateway instance for the target DCI(s), or to know how to work with a corresponding cloud service.

SG instance operators should be IT people who are able to:

- install and configure a SG instance, or know how to work with a corresponding cloud service,
- manage users,
- take care of security issues (for example, manage firewalls, etc.),
- regularly update the gateway and the underlying software stack, or maintain cloud service compatibility.

8.3 Who are these Players in Europe?

There are many SG instance developer communities in Europe. It is not the goal of this document to show all of them rather we show some success stories of gateway instances that already work at production level and serve relatively large communities. The other European gateway developer communities can be found in the EGI Application Database [7.1].

8.3.1 MoSGrid gateway

MoSGrid (Molecular Simulation Grid) focuses on the configuration and provision of Grid services for molecular simulations. MoSGrid shall enable extensive use of D-Grid-Infrastructure for high-performance computing in the field of molecular simulation including annotation of the results with metadata and their provision for data mining and knowledge generation. MoSGrid aims to support the users in all fields of simulation calculations. Via a portlet, the user can access data repositories where information on molecular properties as well as on "recipes" – standard methods for the provided applications – is stored. By means of these recipes simulation jobs can be automatically generated and submitted into the Grid (Preprocessing and Job Submission). Moreover, the users are supported at the analysis of their calculation results. This facilitates the post processing of the data for subsequent calculations and analyses. Through the cross-referencing of different result data sets new insights can be achieved. The data repository additionally allows external referencing of simulation results. In order to achieve these goals MoSGrid has created a SG instance for its user community. The MoSGrid gateway is customized from WS-PGRADE/gUSE. During the customization process for example, DCI Bridge was connected to UNICORE and the SAML authentication mechanism was introduced to WS-PGRADE/gUSE. The advantage of such customization is that now all these new features are part of the recent WS-PGRADE/gUSE releases and hence other communities (not only the MoSGrid community) can take advantage of these new features if they use WS-PGRADE/gUSE.

8.3.2 Swiss Grid Proteomics Portal

Systems biology studies biological systems and processes as dynamic, integrated networks of interacting molecules. The underlying vision is that the structure and dynamics of such networks will provide insights into the function and control of biological systems that are not apparent from studying the systems components. The technologies to collect the data required to analyze molecular networks and the theory and tools to extract functional information from the data is being developed at present in many systems biology research institutions around the globe. By its nature, systems biology is an interdisciplinary science, joining researches from the domains of Biology, Medicine, Chemistry, Bioinformatics, Computer Science, Physics, Mathematics and Engineering.

The Swiss Grid Proteomics Portal is based on the WS-PGRADE/gUSE portal technology. This gives the ability to submit to a large number of Grid systems and to different kinds of remote and local resources. Developers of the gateway instance in ETH Zurich have extended the functionality of WS-PGRADE/gUSE by collaborating closely with the developers of WS-PGRADE/gUSE.

Video demos on the gateway can be found at http://www.sci-bus.eu/demos11.

8.3.3 VisIVO Gatetway

VisIVO Science Gateway aims to create an astrophysical portal based on the generic-purpose **gUSE/WS-PGRADE portal** family to access **VisIVO** software tools. It has been developed in the framework of the **SCI-BUS Project**.

VisIVO is a suite of software tools for creating customized views of 3D renderings from astrophysical data tables. These tools are founded on the **VisIVO Desktop** functionality (visivo.oact.inaf.it) and support the most popular Linux based platforms (e.g. www.ubuntu.com). Their defining

characteristic is that no fixed limits are prescribed regarding the dimensionality of data tables input for processing, thus supporting very large scale datasets.

VisIVO Server websites are currently hosted by the University of Portsmouth, UK (visivo.port.ac.uk), the INAF Astrophysical Observatory of Catania, Italy (visivo.oact.inaf.it) and in the near future by CINECA, Italy (visivo.cineca.it). These web sites offer data management functionality for registered users; datasets can be uploaded for temporary storage and processing for a period of up to two months. The sites can also be utilized through anonymous access in which case datasets can be uploaded and stored for a maximum of four days; to maximize available resources a limited dimensionality is only supported.

Assuming that datasets are uploaded, users are typically presented with tree-like structures (for easy data navigation) containing pointers to **files**, **tables**, **volumes** as well as **visuals**.

Video demos on the gateway can be found at http://www.sci-bus.eu/demos.

8.3.4 Statistical Seismology Science Gateway

The Statistical Seismology Science Gateway (SSS-Gateway) is developed by Middle East Technical University within the framework of the SCI-BUS project. This gateway is a web portal aiming to provide a comprehensive set of statistical methods for e-scientists working in the field of seismology. SSS-Gateway is based on the SCI-BUS generic-purpose gateway technologies, formed by WS PGRADE/gUSE and integrated with the portal framework Liferay, providing a DCI virtualization environment with a scalable set of high-level services. The current version of SSS-Gateway contains the following portlets regarding the Statistical Seismology Functions (SSF) implemented:

- **Simple-use Portlet for SSF1** – Integration of multi-source data for increased reliability and quality
- **Simple-use Portlet for SSF2** – Determination of probability distributions and their input parameters
- **Simple-use Portlet for SSF3** – Robust techniques for the parameter estimation in models and relations
- **Simple-use Portlet for SSF4** – Complex predictive modeling of earthquake phenomena in time-space domains
- **Advanced-use Portlet** for users to code their models in a programming language calling any combination of the statistical seismology functions provided.

Video demos on the gateway can be found at http://www.sci-bus.eu/demos7.

8.3.5 AutoDock Gateway

SZTAKI has built the AutoDock Gateway derived from WS-PGRADE/gUSE using the end-user interface of this SG framework. The end-user interface hides the workflow development facilities of WS-PGRADE/gUSE but enables the access of the internal repository in order to download predefined workflow applications by the end-user scientists. The AutoDock Gateway currently provides three predefined applications. After downloading an application the user can parameterize it and submit to

the EDGeS@home BOINC-based desktop grid system that has about 20.000 registered volunteer host machines to run these applications. See details at: https://autodock-portal.sztaki.hu/

9 STEPS OF BUILDING YOUR SCIENCE GATEWAY

Author(s): Peter Kacsuk

9.1 Create an Exact List of Requirements your SG Should Meet

Here we provide you the major items for your list:

- Target community
  - Focussed in a certain branch of science - you need SG instance specialized for supporting that branch of science.
  - Scientists from various fields (e.g. NGI SG) - a generic purpose SG framework might do.
- Size of the target community
  - Small number of scientists - the scalability requirements is not so important.
  - Large number of scientists - the scalability requirements is very important.
- Supported applications
  - Focussed in one or very small number of predefined applications
    - you need SG instance specialized for supporting that application(s).
    - you need robot certificate.
  - Large set of applications that can be defined by the users
    - you might need SG framework enabling the development of applications
    - you need certificate management
  - Large set of workflow applications that can be defined by the users
    - you might need SG framework supporting workflow creation and management,
    - you need certificate management,
    - support for SHIWA workflow repository access would be useful.
- Target DCIs
  - Focussed on one particular DCI.
  - Enabling access to several different DCIs (e.g. an NGI where both clouds and different grids are supported) - your SG should be derived from a SG framework that can support many different DCIs.
- Generic requirements
  - Robustness
    - You want only a prototype of your SG - this is not an issue for you, you can develop your SG from scratch.
    - You want to create a production SG service - Choose a robust SG Framework and create your SG instance with customization of the selected SG Framework (if the target SG is extremely simple you can build it from scratch, too).

- Supported functionalities
  - Limited functionalities - you can try to develop your SG instance from scratch
  - Rich functionalities - choose a SG Framework with rich functionalities and create your SG instance with customization of the selected SG Framework
- Sustainability
  - You want to support the target user community only for the time of a project (2-3 years) - you can try to develop your SG instance from scratch.
  - You want to support the target user community for a long term beyond the development project life-time - choose an open source based SG Framework behind which there is a large and active development and user community (this gives a chance that this SG Framework and your customized SG instance will be sustainable for a long term).
  - Decision if you would like to offer pay-per-use services at some point, for all or just for external or commercial users, and you thus need corresponding accounting and invoicing.
- Extensibility
  - You are sure that you do not want to further develop your SG even if its users ask for it – any.
  - You assume that sooner or later your SG should be extended according to the new user needs - choose a SG Framework with layered and SOA architecture that is easily extendible and create your SG instance with customization of the selected SG Framework.
- Scalability
  - Your target community is small and you know that it will not grow in the future - scalability is not an issue for you.
  - Your target community either large or will grow to be large - choose a SG Framework with scalable architecture and create your SG instance with customization of the selected SG Framework.

9.2 Choose Technologies based on Resources and Time

Before you start to build your SG it is very important to carefully plan the use of your available resources and time.

- Small number of person months (PMS) and short deadline - because of the small number of PMs is better to use or customize an existing SG framework.
- Small number of PMS and long deadline - because of the small number of PMs it is better to use or customize an existing SG framework.
- Large number of PMS and short deadline - because of the short deadline it is better to use or customize an existing SG framework.
- Large number of PMS and long deadline - you can develop your own SG instance from scratch if you like but consider the disadvantages mentioned in previous chapters and sections. You can even develop your own SG framework but it is risky.

9.3 Building Portals from Reusable Components

The Liferay technology enables quickly building portals using the Liferay portal framework and the existing Liferay portlets. There is a public Liferay portlet repository where you can find a large set of ready-to-use portlets. These help to build simple SG instances from scratch. The problem with such generic portal frameworks is that they do not provide back-ends that support DCI access. Creating the back-end with the necessary DCI access could be time-consuming and there are already existing SG frameworks and services that can be used for this purpose. E.g. the DCI Bridge service developed in the SHIWA and SCI-BUS projects can be easily connected to any gateway via a standard BES interface and it provides access to a large set of different DCIs (clusters, grids, desktop grids, clouds).

It would be ideal to create a European SG portlet and service repository that contains portlets and services from which as building boxes SG developers could put together their desired SG instance. There are already such reusable portlets and services but the set of them is by far not complete and their integration methods are not crystallized yet.

If you build a SG instance currently there are two options:

- **Build from scratch:** use standard portal building technologies (e.g. Liferay) that guarantee an initial framework and a large set of existing portlets. Use these portlets whenever it is possible and develop only those portlets that specialized for your needs.
- **Customize an existing SG framework:** usually they are built on top of Liferay so they have the same advantages as writing the SG directly on top of Liferay but they provide already many, useful additional functionalities compared to Liferay, portlets and back-end with access to various DCIs.

If you build a SG framework currently there are two options:

- **Build from scratch:** use standard portal building technologies (e.g. Liferay) that guarantee an initial framework and a large set of existing portlets. Use these portlets whenever it is possible and develop only those portlets that specialized for your needs.
- **Customize an existing open source SG framework:** usually they are built on top of Liferay so they have the same advantages as writing the SG directly on top of Liferay but they provide already many, useful further functionalities, portlets and back-end with access to various DCIs. However, it is better to collaborate with the developer community of the selected open source SG framework and jointly further develop that open source SG framework.

9.4 Select a Development Team with User Interface Experience

9.5 Plan for the Long Term

If your SG should be maintained for a long time it is very important to consider this requirement from the very beginning of designing your SG. This is a typically overlooked aspect of designing and establishing SGs. Many projects consider only the lifetime of their project and they do not care what will happen to the project products (including the developed SG) after the project is completed. As a result many SGs become unusable after the developing project is over. To avoid this pitfall the following recommendations should be followed:

- If there is any existing open source SG framework that can be customized to your need always develop your SG instance with SG framework customization and do not start to develop it from scratch:
    - customization needs less development and maintenance cost than scratch-development,
    - the open source community behind the SG framework has got chance to enforce the required future costs for sustainability.
- If your user community is small do not put much effort to the development of the SG instance: a small community is not able to enforce the required future costs for sustainability.
- Try to enlarge your user community: a large user community has got chance to enforce the required future costs for sustainability.
- Using cloud services and thus outsourcing a large part of the operations and maintenance might be an attractive possibility for you.

9.6 Develop in Stages

It is very difficult to develop and provide exactly that kind of SG that your community wants. Instead of putting many efforts to find out completely what the user community wants it is better to build a simple SG instance for them with small development effort. Then provide the SG service to the user community and ask for feedback.

It is important that even this simple SG instance should be robust and user friendly, otherwise the user community will be disappointed with the service provided and look for alternatives. If the user community likes the simple SG then they will tell you what to improve and extend. Based on this feedback you can enter into the second stage of SG development. At this point it is very important that even the simple SG should be scalable and extensible. These features enable the fast extension of the SG and the fast growth of the number of the users. Do not put too many new services in the new version of the gateway. Every new version should provide some clear new functionality and the new version should be as robust as the previous one was, otherwise users will not come back to your SG.

In summary, providing frequently small extensions as new releases is a better practice than rarely providing a new release with lots of extensions. In general, applying agile software development procedures might be very useful here.

10 INTEGRATION WITH EGI INFRASTRUCTURE

Author(s): Nuno Loureiro-Ferreira, Dusan Vudragovic, Petar Jovanovic, Antun Balaz

10.1 EGI.eu Policies

10.1.1 Overview

EGI.eu policies are clear, formal and mandatory statements or positions adopted by the EGI.eu governance bodies for issues relevant to the EGI community. EGI.eu policies are important to provide strategic guidance, improve decision making processes, clarify roles and responsibilities amongst all actors and provide help in managing risks. The policies cover a wide range of issues, from long-term strategic cooperation with other e-infrastructure providers, to specific problems affecting EGI user communities. The successful implementation of EGI.eu policies within a well-defined strategic framework will contribute to the main goal of the EGI.eu and European Grid community in general.

Policies and procedures are developed either internally by policy groups or in collaboration with external partners. The policy development process rests on the fundamental principles of openness, transparency and consensus. More information can be found at the EGI.eu Strategy and Policy Team (EGI.eu SPT) webpages [10.1].

Relevant policies mostly related to gateway development are briefly described in the next section. The comprehensive list of approved policies and procedures is available for consultation at the EGI.eu SPT webpages.

10.1.2 Policies approved and in use by the EGI community

Grid Security Policy [10.2]

Policy Statement

This document presents the Policy regulating those activities of Grid participants related to the security of Grid services and resources.

Virtual Organization Portal Policy [10.3]

Policy Statement

This Policy applies to all Portals operated by Virtual Organisations that participate in the Grid.

Service Operations Security Policy [10.4]

Policy Statement

This security policy presents the conditions that apply to anyone running a Service on the Infrastructure, or to anyone providing a Service that is part of the Infrastructure. This policy is effective from 1st February 2012 and replaces two earlier security policies, namely the Grid Site Operations Policy [10.5] and the Site Registration Security Policy [10.6].

Grid Security Traceability and Logging Policy [10.7]

Policy Statement

This policy defines the minimum requirements for traceability of actions on Grid Resources and Services as well as the production and retention of security related logging in the Grid.

Security Incident Response Policy [10.8]

Policy Statement

This document describes the policy and responsibilities for handling security incidents affecting the Grid.

Policy on Grid Multi-User Pilot Jobs [10.9]

Policy Statement

Security policy for operation of multi-user pilot jobs.

Grid Policy on the Handling of User-Level Job Accounting Data [10.10]

Policy Statement

This document presents the minimum requirements and policy framework for the handling of user-level accounting data created, stored, transmitted, processed and analysed as a result of the execution of jobs on the Grid.

For further information on all current EGI policies and procedures in place, check the following reference [10.11].

10.2 How to Integrate Portals & Enabling Technologies with EGI AppDB

10.2.1 Overview

Access to the European Grid Infrastructure (EGI) is possible through a range of different service interfaces. Science gateways are emerging as promising tools offering a collaborative environment and enabling a seamless access to computing and data storage resources, helping scientists to engage more actively with the underlying e-infrastructure.

The EGI.eu User Community Support Team together with its partners from the National Grid Infrastructures and Virtual Research Communities setup dedicated webpages to present basic information about ready to use science gateways [10.12].and about science gateway enabling technologies [10.13].for gateway developers. EGI science gateways and their enabling technologies are registered in the EGI Applications Database.

10.2.2 Applications Database in a nutshell

The Applications Database (AppDB) [7.1] is a service that stores information about tailor-made scientific software tools, as well as the profiles of the programmers and scientists who developed and make use of them. The software in AppDB is ready to be used on the European Grid Infrastructure and other European Distributed Computing Infrastructures (DCIs). Having a searchable catalogue of reusable applications and developer tools means, that scientific communities should spend less time developing and integrating software to DCIs while devoting their time on delivering science. AppDB helps the communities to identify related and reusable services, modules, and frameworks, whether they want to develop a new algorithm or want to run a widely used application on an e-infrastructure. The database has been online since 1 July 2010 and is the natural successor to the Enabling Grids for e-science database [10.14]. The overarching goal of EGI is to make AppDB a sustainable community-driven database for the community. AppDB is an open service, providing read-only access to everyone, while users with an EGI SSO account [add ref to url] can take advantage of the authenticated write-enabled features of the portal. Requests for new AppDB features can be fed through the EGI RT system [add ref to url] and bug reports through the EGI Helpdesk [add ref to url].

10.2.3 AppDB relevant features

The Applications Database has a rich set of features made available to the community. Under the scope of this Primer, the most relevant features available to enablers/developers of Science gateways will be briefly highlighted. The main features abridged touch topics such as mechanisms available for users to interact with the database and respective entries, interoperability, quality and dissemination of information. The complete list of features is rather extensive and can be consulted in the AppDB release log [add ref to url].

Usage

- Search/Advanced search

  Search boxes throughout the portal feature semantics enabled syntactic searching, meaning that they take filter expressions as search arguments which will be matched to results that are deemed relevant in the scope of the search target. Further information can be found in the AppDB FAQ [add ref to url].

- Associate people and software

  Each software entry in AppDB is associated to a set of profiles, possibly the developers of the scientists that ported the software to the grid.

- Add contact person expertise

The profile of the contact persons for the software entries can be further highlighted by providing more specific details on their expertise.

- Application Tagging

  Tags can be attached to any of the stored items. Tag-based search and browse capabilities were added to the system.

User communication

- Join as a contact point

  This feature allows new contacts to be added to current AppDB software entries.

- Rating & comments

  AppDB allows community members to provide feedback and to rate any of the stored items. This feature greatly simplifies the central content management team to correct and revoke any AppDB entry profile flagged as inaccurate or inappropriate. Reviews and ratings, associated with number of web visits per AppDB registry, will help the visitor to identify the most relevant entries.

- Send a message to a person associated with a given software item

  An AppDB registered member can engage directly with any contact point of a certain software through this feature.

Interoperability

- REST API v1.0 with write support

  A REST API that supports authenticated writes and updates of the database is available. Third party software providers can make use of the API by forwarding their users' EGI SSO credentials, or by creating an AppDB system account to act on behalf of their users, in order to modify content and to read content that is not open to the general public. Authenticated access is granted based on permissions readily assigned to the account identified by the credentials passed. API documentation with examples and sample use cases is available in the EGI Wiki [add ref to url].

Quality of information

- Broken link detection tool

  A 'broken link detection engine' is already in production. As the name implies it monitors all hyperlinks associated to AppDB profiles. The broken link notification system automatically sends e-mail notifications and reminders to the owners of AppDB registries, if associated links are inaccessible for a period of over 3 weeks. An action needs to be taken to update the hyperlinks otherwise the AppDB entry is tagged as inaccurate.

- Multiple application categories & extended categorization-based classification model

  A mechanism enabling the community to classify the registered software under multiple high level categories. The 'Science Gateways' category was already added to AppDB as a main category [add ref to url]. The list of categories that the community would like to see in the system is under discussion, but the tool to implement new categories is already in place.

- Error & problem reporting
- Supports a review process

  Authorized users can flag content. End-users will not be able to see it until the review process is finished by AppDB team.
- Mechanism to keep entries up-to-date

  A mechanism that identifies software entries that have not been updated in the last 12 months. These entries are visually tagged with a small emblem notifying users that the content about the software may not be up to date. Users can optionally exclude such 'probably outdated' applications from their searches.

Dissemination of information

- Dissemination tool

  A dissemination/outreach feature is in place to all AppDB members with appropriate permissions [add ref to url]. The feature will allow broadcasting, through email, customized groups of users registered in the AppDB portal.
- News through Email subscriptions and RSS feeds

  A notification system has been integrated into AppDB to allow subscription for email notifications or RSS feeds. This feature aims to simplify the task of knowing what's happening for example within your scientific domain or country.
- Notifications about software updates to owners and related contacts

10.2.4 How to register an EGI science gateway in AppDB

EGI science gateways [add ref to url] and their enabling technologies are registered in the EGI Applications Database. The Applications Database stores key information about the gateways and links to external pages providing additional information about them (e.g. access page, download page, available publications).

By registering your science gateway or gateway enabling technology in the Applications Database your service will be visible to the EGI community and to the general public, not only through the EGI website but also through numerous user community and National Grid Initiative websites [add ref to url] that use the Applications Database web gadget. Through this web-gadget you can also embed the list of EGI gateways or gateway enabling technologies into your own website.

To register your EGI science gateway or gateway enabling technology in the database, proceed as follows:

1. Log in with your EGI SSO account.

   If you don't have a SSO account, please register as a new user [add ref to url]. It's free and it will allow you to access most EGI-related web-based tools.
2. Login to the EGI Applications Database [add ref to url] and click on the 'Register New' page.
3. Register it as a 'science gateway' (set it as primary)
   You may choose to add more categories to your software entry, but keep 'science gateways' as the primary category for proper AppDB handling.

4. If you want to **register a web-based science gateway**, then:
    a. Provide a 'Try it' link, so those who wish can access your gateway
    b. Tag it with *web.gateway*
5. If you want to **register a desktop application-based gateway**, then:
    a. Provide a download link, so those who wish can install your gateway
    b. Tag it with *application.gateway*
6. If you want to **register a science gateway enabling technology**, then:
    a. Provide a download link, so those who wish can access your technology
    b. Tag it with *technology.gateway*

If you have any further questions send an email to EGI.eu User Community Support Team.

10.3 How to Integrate Portals with EGI Monitoring System

Scientific Gateways (SG) have become an essential tool for research in the age of e-Science. Their operation and performance needs to be monitored in order to ensure quality of service for end users. Such monitoring also has to provide an integrated overview of the global status of the scientific gateways, as well as the detailed status of the individual scientific gateway layers and components. In addition, it has to enable sending of alerts to administrators when a particular issue is identified, to enable scheduling of downtimes during service maintenance, as well as to produce statistical reports on infrastructure performance.

Within the EGI-InSPIRE project, detection and diagnose of the operational problems is accomplished through several monitoring tools: GOC DB, GStat, SAM framework, Real Time Monitoring, GridView, Google Earth, and GridMap. On top of these, the Operations Dashboard provides links and utilizes combined views to simplify monitoring tasks.

GOC DB [https://goc.egi.eu/] is the central static information repository, and SGs will have to be registered there so as to enable their automated monitoring. GOC DB stores information about NGIs, Grid sites, services provided by the Grid sites, users, etc. It is commonly used by the Grid site administrators to declare maintenance for (un)scheduled events. GOC DB architecture [https://wiki.egi.eu/wiki/GOCDB] consists of three main parts: a database – where all information is stored; a web portal – which interfaces with the database; and a programmatic interface – which exports initial configuration for the Grid information system.

The introduction of SGs into the GOC DB implies creation of a new GOC DB object. This object has to contain common attributes that will describe any SG, such as: name of the portal, portal URL, type of the portal (used technology), version of the portal, contact persons (administrators, user support, security, etc.), available applications, etc. In addition to these attributes, in order to allow SG instances to dynamically publish information to the Grid Information System, an SG LDAP URL has to be specified.

Current implementation of the Grid Information System relies on OpenLDAP [T. Jackiewicz, "Deploying OpenLDAP" (Springer-Verlag, New York, 2004)], an open source implementation of the Lightweight Directory Access Protocol (LDAP). All Grid services publish information about themselves via a service called resource-level BDII. The resource-level BDII runs what is known as information provider to obtain relevant information about the particular instance of a Grid service and caches the

result in an OpenLDAP database. The site-level BDII gather information about all Grid services running at that site by querying all the resource-level BDIIs, and caches the results in its OpenLDAP database. Finally, a top-level BDII queries all the site-level BDIIs that exist in the Grid infrastructure, defined by the configuration file exported from the GOC DB. Taking this into account, we see that integration of an SG into the Grid Information System requires installation of an OpenLDAP server (resource-level BDII) per SG, which will enable provision of both static and dynamic SG properties to the Grid Information System. Using this resource-level BDII, SG will be able to publish real-time information such as: number of total, running, waiting jobs, number of jobs per application, number of available job slots, available applications, etc. When this is realized, various Grid Information System-based monitoring tools, like GStat [https://wiki.egi.eu/wiki/External_tools#Gstat][http://gstat.egi.eu/], Google Earth or Grid Map [http://grid-monitoring.cern.ch/myegi/gridmap/], will be able to automatically provide monitoring of the SGs. In particular, GStat will be able to detect faults in SG information systems, to verify the validity of the information, and to give visualisation of the numerical properties. Google Earth, in conjunction with the information coming from GOC DB export system, will be able to generate maps for geographical distribution of SGs. And finally, Grid Map will be able to present graphical representation of the SG CPU power and its availability.

Central EGI-InSPIRE monitoring framework is the Service Availability Monitoring [https://wiki.egi.eu/wiki/SAM] (SAM). This system provides status and history of all services and sites within the infrastructure, visualization of services and sites' availabilities, and web services for data exports. It relies on the existing technologies: Nagios [W. Barth, "Nagios: System and Network Monitoring" (Oreilly & Associates, 2006)] and ActiveMQ [B. Snyder, R. Davies and D. Bosanac, "ActiveMQ in Action" (Manning Publications, 2011)]. Nagios is used for scheduling and execution of probes, while the ActiveMQ (MSG messaging system) integrates other operational tools with Nagios instances. The SAM framework is a distributed monitoring tool, and uses three central databases: Aggregated Topology Provider [https://tomtools.cern.ch/confluence/display/SAM/ATP] (ATP), Metric Description Database (MDDB), and Metric Results Store [https://tomtools.cern.ch/confluence/display/SAM/MRS] (MRS), which centrally store distributed information for project monitoring purposes.

An essential part of the SAM framework is the Nagios Config Generator [https://tomtools.cern.ch/confluence/display/SAM/NCG] (NCG). This tool enables automatic generation of a Nagios configuration, based on multiple information sources, but mainly on GOC DB and Grid Information System. Practically, this means that, once the SG has a working resource-BDII server (LDAP server) and announces this information through the GOC DB to the top-level BDIIs, NCG will be able, using this information, to generate configuration for the Nagios server that will monitor this particular SG, or SGs within the NGI, or all SGs within the infrastructure.

The SAM framework uses the Nagios technology during the last several years, and a set of guidelines and know-hows for the development of Grid Nagios probes are already available. These guides cover several of the most common cases: development of a simple probe, a multi-test probe, and a long-running probe. Single probes test a specific service in a single run (testing period is short). A multi-test probe in a single run performs multiple tests using combination of active and passive checks, while the long-running probe wraps Grid job submission, monitoring of the submitted job, and the retrieval of the job result. All these probe-types could find their place in the monitoring of SGs. For instance, single probes could be used for portal availability checks, multi-test probes for SG local

storage checks (store, retrieve, and delete files), while the long-running probes could investigate a complete SG job life-cycle.

A generic SG architecture consists of several layers illustrated in [Figure]: Presentation layer – which is usually realised as a web portal; Middle layer – which contains workflow engine, SG information system, SG application repository, etc.; and Architecture layer – which is responsible for job submission to different DCIs. In the most generic case, all three layers are open for end-users. The presentation layer is usually used by non-experienced users, or users that just started porting of their application to the DCI. The middle layer could be interesting for the developers that make use of the running application portal, which is isolated from any batch system and workflow engines, and thus only the server machine itself is used for execution of applications. Finally, the architecture layer could be used by other portals and applications that have implemented the workflow engine capability, but the execution is limited to the local cluster. In such cases, the application will benefit from the available computation power provided by different DCIs and accessible through the SG.

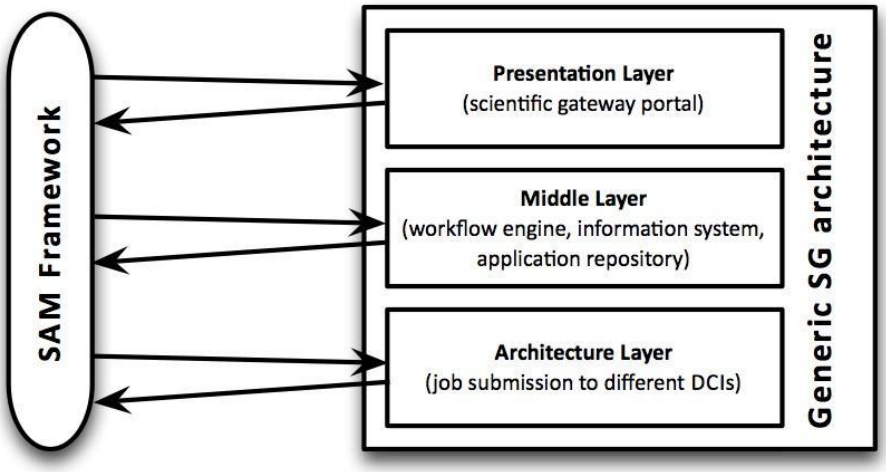

Figure 3 *Monitoring of the Generic Scientific Gateway*

The SAM framework has to enable monitoring of all SG layers and components in order to ensure quality of service for end users. Nagios presentation layer probes will have to be developed to check availability of the portal and its components, portal authentication mechanism, the management of input data, workflow and data-flow tools, and application submission. Middle Layer probes will be responsible for validation of the portal application repository, workflow storage and interpreter, as well as portal local file storage. Finally, Nagios long-running probes will be used to validate submission to different DCIs (gLite, ARC, UNICORE, Globus, LFS, PBS, BOINC, web service, local resource, Google App Engine, etc.) through the architecture layer.

Visually the most attractive EGI monitoring tool is the Real Time Monitoring [http://rtm.hep.ph.ic.ac.uk/] (RTM). This real-time monitor overlays Grid activity onto a 3D globe, representing Grid sites with pulsing circles, WMS services with triangles, and on-going job activities

with lines. SGs have to find their place on RTM maps, which are presented to various audiences, and thus will provide an attractive dissemination tool for SG Grid activities.

Finally, to simplify monitoring tasks and to provide combined views, as well as to enable sharing and dissemination of information and discussions within the EGI community, SGs have to become integral part of the Operations Dashboard [https://wiki.egi.eu/wiki/Operations_Portal] [https://operations-portal.egi.eu/]. Such development will require filtering of messages related to SGs from the SAM framework's MSG messaging system, and their presentation in the form suitable for the operations teams. In addition to the existing COD and VO dashboards, a new SG dashboard should be introduced. It should give an overview of all detected problems related to SGs, and enable operations staff to track them. Quite popular VO info feature from the current Operations Dashboard could be extended to include information on SGs and how to support them (offer resources), thus promoting the use and support for SGs. Ultimately, Operations Dashboard broadcast system should be extended as to allow dissemination of information and announcements related to SGs (new version, new application, new feature, etc.).

11 REFERENCES

[2.1] Z. Farkas and P. Kacsuk: P-GRADE Portal: a generic workflow system to support user communities, Future Generation Computer Systems journal, Volume: 27, Issue: 5, 2011, pp. 454-465.

[2.2] R. Barbera, A. Falzone, V. Ardizzone, D. Scardaci, "The GENIUS Grid Portal: Its Architecture, Improvements of Features, and New Implementations about Authentication and Authorization," 16th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE 2007), Paris, pp.279-283, 2007.

[2.3] Dahan M, Boisseau J R. The GridPort Toolkit: A System for Building Grid Portals. Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing 2001:216.

[2.4] Krzysztof Kurowski, Piotr Dziubecki, Piotr Grabowski, Michał Krysiński, Tomasz Kuczyński and Dawid Szejnfeld: Easy development and integration of Science Gateways with Vine Toolkit, Journal of Grid Computing, Vol. 10, No. 4, 2012.

[2.5] Peter Kacsuk, Zoltan Farkas, Miklos Kozlovszky, Gabor Hermann, Akos Balasko, Krisztian Karoczkai and Istvan Marton: WS-PGRADE/gUSE generic DCI gateway framework for a large variety of user communities, Journal of Grid Computing, Vol. 10, No. 4, 2012.

[2.6] Sciacca E., Bandieramonte M., Becciani U., Costa A., Krokos M., Massimino P., Petta C., Pistagna C., Riggi S., and Vitello F. (2013).

"VisIVO Workflow-Oriented Science Gateway for Astrophysical Visualization". Accepted to be presented at Parallel, Distributed, and Network-Based Processing PDP 2013 and will be published by IEEE Computer Society, Conference Publishing Services (CPS).

[2.7] P. Kunszt, L. Malmström, N. Fantini, W. Sudholt, M. Lautenschlager, R. Reifler, S. Ruckstuhl: Accelerating 3D protein modeling using cloud computing, Workshop on Computing Advances in Life Science, 7th IEEE International Conference on eScience, Stockholm, December 5-8, 2011, IEEE Computer Society.

[2.8] T. Kiss, P. Greenwell, H. Heindl, G. Terstyanszky, and N. Weingarten. Parameter sweep workows for modelling carbohydrate recognition. Journal of Grid Computing, Vol. 8, No. 4, pp 587-601, 2010.

[3.1] T Delaitre, T Kiss, A Goyeneche, G Terstyanszky, S Winter, P Kacsuk: GEMLCA: Running legacy code applications as grid services, Journal of Grid Computing 3 (1), 75-90, 2005.

[3.2] Radu Prodan Specification and runtime workflow support in the ASKALON Grid environment, Scientific programming, vol 15, no 4, 2007. IOS Press.

[3.3] Ludäscher B., Altintas I., Berkley C., Higgins D., Jaeger-Frank E., Jones M., Lee E., Tao J., Zhao Y. 2006. Scientific Workflow Management and the Kepler System. Special Issue: Workflow in Grid Systems. Concurrency and Computation: Practice & Experience 18(10): 1039-1065.

[3.4] Johan Montagnat et al: A data-driven workflow language for grids based on array programming principles, in proceedings of "Workshop on Workflows in Support of Large-Scale Science (WORKS'09), Portland, USA, 2009.

[3.5] Tom Oinn et al: Taverna: a tool for the composition and enactment of bioinformatics workflows, Bioinformatics, Vol. 20 no. 17, pages 3045–3054, 2004.

[3.6] Matthew Shields and Ian Taylor: Programming Scientific and Distributed Workflow with Triana Services.

[3.7] P. Kacsuk, K. Karóczkai, G. Hermann, G. Sipos, J. Kovács: WS-PGRADE: Supporting parameter sweep applications in workflows. In: 3rd Workshop on Workflows in Support of Large-Scale Science, In conjunction with SC 2008, Austin, TX, USA, 2008.

[3.8] DCI Bridge: http://sourceforge.net/projects/dcibridge/

[3.9] Tom Goodale, Shantenu Jha, Harmut Kaiser, Thilo Kielmann, Pascal K Leijer, Gregor von Laszewski, Craig Lee, Andre Merzky, Hrabri Rajic, John Shalf: SAGA: A Simple API for Grid applications, High-Level Application Programming on the Grid Computational Methods in Science and Technology, vol. 12, No. 1, 2006.

[3.10] CloudBroker Platform - http://www.cloudbroker.com/index.php/products

[3.11] gLite WMS: F Pacini, EGEE User's Guide, WMS Service, DATAMAT, 2005.

[3.12] CREAM - http://grid.pd.infn.it/cream/field.php

[3.13] Tristan Glatard and Sorina Camarasu-Pop: Modelling pilot-job applications on production grids.

[3.14] J. T. Mosciki: Distributed analysis environment for HEP and interdisciplinary applications. Nuclear Instruments and Methods in Physics Research A, 502:426429, 2003.

[3.15] Adrian Casajus, Ricardo Graciani, Stuart Paterson and Andrei Tsaregorodtsev: DIRAC pilot framework and the DIRAC Workload Management System, Journal of Physics: Conference Series Volume 219 Part 6, 2010.

[4.1] Luis M. Vaquero, Luis Rodero-Merino, Juan Caceres, Maik Lindner, “A Break in the Clouds: Towards a Cloud Definition”, ACM SIGCOMM Computer Communication Review, Volume 39, Issue 1, January 2009, Pages 50-55.

[4.2] Peter Mell, Timothy Grance, “The NIST Definition of Cloud Computing”, NIST Special Publication 800-145, September 2011.

[4.3] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy H. Katz, Andrew Konwinski, Gunho Lee, David A. Patterson, Ariel Rabkin, Ion Stoica, Matei Zaharia, “Above the Clouds: A Berkeley View of Cloud Computing”, Electrical Engineering and Computer Sciences University of California at Berkeley, Technical Report No. UCB/EECS-2009-28, February 10, 2009.

[4.4] Maryline Lengert, Bob Jones, “Strategic Plan for a Scientific Cloud Computing Infrastructure for Europe”, CERN-OPEN-2011-036, August 8, 2011.

[4.5] Katherine Yelick, Susan Coghlan, Brent Draney, Richard Shane Canon, Lavanya Ramakrishnan, Adam Scovel, Iwona Sakrejda, Anping Liu, Scott Campbell, Piotr T. Zbiegiel, Tina Declerck, Paul Rich,

et al., “The Magellan Report on Cloud Computing for Science”, U.S. Department of Energy, Office of Advanced Scientific Computing Research (ASCR), December 2011.

[7.1] EGI Applications Database: http://appdb.egi.eu/

[10.1] EGI.eu Strategy and Policy Team - http://www.egi.eu/about/policy/

[10.2] Grid Security Policy - https://documents.egi.eu/document/86

[10.3] Virtual Organization Portal Policy - https://documents.egi.eu/document/80

[10.4] Service Operations Security Policy - https://documents.egi.eu/document/669

[10.5] Grid Site Operations Policy - https://documents.egi.eu/document/75

[10.6] Site Registration Security Policy - https://documents.egi.eu/document/76

[10.7] Grid Security Traceability and Logging Policy - https://documents.egi.eu/document/81

[10.8] Security Incident Response Policy - https://documents.egi.eu/document/82

[10.9] Policy on Grid Multi-User Pilot Jobs - https://documents.egi.eu/document/84

[10.10] Grid Policy on the Handling of User-Level Job Accounting Data - https://documents.egi.eu/document/85

[10.11] European Grid Infrastructure: Policies and Procedures - http://www.egi.eu/about/policy/policies_procedures.html

[10.12] European Grid Infrastructure: Science Gateways for Users - http://www.egi.eu/services/support/science-gateways/science_gateways_for_users.html

[10.13] European Grid Infrastructure: Science Gateways for Developers - http://www.egi.eu/services/support/science-gateways/science_gateways_for_developers.html

[10.14]

[10.15]

[10.16]

[10.17]

[10.18]

[10.19]

[10.20]

[10.21]

[10.22]

12 ABOUT THE AUTHORS

Balaz, Antun is an Associate Research Professor at the Institute of Physics Belgrade. He is technical coordinator of AEGIS, and director-designate of Blue Danube National Supercomputing and Data Storage Center. Dr. Balaz was operations (SA1) leader in SEE-GRID-2 and SEE-GRID-SCI projects, and serves as Serbia country representative in EGI-InSPIRE, PRACE-1IP, PRACE-2IP, PRACE-3IP, and HP-SEE. Research interests of Dr. Balaz are in devising efficient HPC algorithms for calculation of path integrals and their applications in high-energy and condensed matter physics, including Bose-Einstein condensation and disordered systems. He has more than 60 publications in per-reviewed scientific journals and conference proceedings.

Cauhé, Elisa has a background in Computer Science at the University of Zaragoza. She has been working in the Institute for Biocomputation and Physics of Complex Systems (BIFI) since 2008 in the areas of computation, dissemination and currently as project manager. Especially notable was her role in the creation of the Ibercivis Foundation, the Latin American main initiative in volunteer computing where she is the secretary. She is also involved in projects of collective intelligence, grid and cloud, complex networks and citizen science.

Chen, Hsin-Yen is a senior manager of Academia Sinica Grid Computing (ASGC) since 2006. He is responsible for the development of the e-Science applications on the worldwide distributed computing environment. He is involving in the drug discovery, biomedical, earthquake, tsunami and weather simulation applications developing on the gLite grid environment. There are a web-based high throughput gird enabled virtual screening service (GVSS) and weather simulation (gWRF) developed and deployed on service grid and desktop grid.

Jovanovic, Petar is a Research Assistant at the Institute of Physics Belgrade, working toward his master thesis in software engineering at the University of Belgrade. He provides user support within the Serbian NGI AEGIS, as well as user support and HPC application porting within the Serbian PRACE team.

Kacsuk, Peter is the Director of the Laboratory of the Parallel and Distributed Systems in MTA SZTAKI. He has been leading the P-GRADE and recently the WS-PGRADE/gUSE gateway framework development project since 2003. He is the coordinator of the EU FP7 SCI-BUS project that further develops WS-PGRADE/gUSE gateway framework and

develops 27 science gateway instances based on WS-PGRADE/gUSE gateway framework. He is also co-editor-in-chief of the Journal of Grid Computing published by Springer.

Loureiro-Ferreira, Nuno has a background in Biochemistry and holds a PhD in Biological Chemistry from the University of Coimbra. Protein structural biology and molecular modelling were his key areas of research. Before joining EGI.eu, Nuno worked for the e-NMR consortium at Utrecht University, a grid-enabled e-infrastructure providing the bio-NMR community with a platform integrating and streamlining computational approaches. Site grid administration, virtual organisation software manager and user support officer were his main roles.

Lovas, Robert is the deputy head of the Laboratory of the Parallel and Distributed Systems at the Computer and Automation Research Institute, Hungarian Academy of Sciences (MTA SZTAKI). He received his MSc and PhD degrees at the Budapest University of Technology and Economics. He has long-running experience in ICT collaborations with academic organizations, universities, and enterprises on various application areas of meteorology, biotechnology, telecommunication, and computational chemistry. He has been coordinating FP7 e-Infrastructure projects (DEGISCO and IDGF-SP) and acting as an International Liaison of the Hungarian National Grid Initiative in the EGI-Inspire project. He is a co-author of more than 40 scientific papers and book chapters on parallel software engineering and Grid computing particularly from design, debugging, and application aspects.

Olabarriaga, Silvia D. is assistant professor at the Academical Medical Center (AMC), Amsterdam. She has studied computer science and obtained a PhD in the field of Medical Imaging. Since 2005 her research interests lie on e-science, in particular on improving the mechanisms to exploit e-infrastructures for biomedical research. She leads the e-bioscience research line of the AMC (www.bioinformaticslaboratory.nl) and participates in several national and international initiatives in the field of e-science.

Shahand, Shayan is a PhD candidate at the Academic Medical Center (AMC) of the University of Amsterdam, Netherlands. He obtained his master of engineering degree from the School of Computer Engineering of the Nanyang Technological University, Singapore. His current research interests include the design, development and deployment of front-ends to biomedical data analysis on e-infrastructures, in particular to facilitate adoption of large distributed systems.

Sudholt, Wibke is CTO and Managing Partner of CloudBroker GmbH, a spin-off company of the ETH Zurich in Switzerland that provides solutions for scientific and technical applications in the cloud. She also leads a work package in the SCI-BUS EU FP7 project. Wibke Sudholt obtained chemistry diploma and doctoral degree in Germany and performed postdoctoral research in the US and Switzerland. As project leader for grid computing at the University of Zurich she participated in national and international grid computing projects. She was also Executive Board member of the Swiss National Grid Association (SwiNG) and of EuroCloud Swiss. End of 2008, she co-founded CloudBroker, where

she is now responsible for technical development, operations and administration. Wibke Sudholt has several years of experience in software development, distributed, cluster, grid and cloud computing and associated application porting and enabling, scientific workflows, as well as in the management of corresponding projects and people. Her domain knowledge reaches from quantum chemistry and molecular dynamics simulations to bioinformatics and insurance calculations.

Vudragovic, Dusan is a Research Assistant at the Institute of Physics Belgrade, working toward his PhD thesis in physics at the University of Belgrade. He is involved in Grid operations, where he has extensive experience and currently serves as a deputy manager of Serbian NGI AEGIS. He has also developed a number of operational and monitoring tools within SEE-GRID and HP-SEE series of projects, which are deployed within the SEE region.