GFD-I.180
OGF – Production Grid Infrastructure

http://forge.gridforum.org/sf/projects/pgi-wg

Editors:
Morris Riedel, JSC/FZJ
Johannes Watzl, LMU

20 March, 2011

OGF – Production Grid Infrastructure
Use Case Collection
Version 1.0

Category: INFORMATIONAL

Status of This Document

This document provides information about the production use cases of planned open standard specifications for production Grid infrastructures. It does not define any standards, requirements, or technical recommendations. Distribution is unlimited.

Copyright Notice

Abstract

The Production Grid Infrastructure (PGI) working group works on a well-defined set of standard profiles, and additional standard specifications if needed, for job and data management that are aligned with a Grid security and information model that addresses the needs of production Grid infrastructures. These needs have been identified in various international endeavors and are in many cases based on lessons learned obtained from the numerous activities in the Grid Interoperation Now (GIN) community group. Therefore, PGI can be considered as a spin-off activity of the GIN group in order to feed back any experience of using early versions of open standards (e.g. BES, JSDL, SRM, GLUE2, UR, etc.) in Grid production setups to improve the standards wherever possible. This particular document is a survey of common use cases provided by different stakeholders of PGI profiles or standard specifications. Such stakeholders include production Grid and e-science infrastructures as well as technology providers. The goal of this document is to have a foundation for a set of important requirements to be addressed by the PGI set of profiles and/or specifications.

Contents

1. Introduction

The objective of the Production Grid Infrastructure (PGI) working group is to formulate a well-defined set of profiles, and additional specifications if needed, for job and data management that are aligned with a Grid security and information model that addresses the needs of production Grid infrastructures. These needs have been identified in various international endeavors and are in many cases based on lessons learned obtained from the numerous activities in the Grid Interoperation Now (GIN) community group. Therefore, PGI can be considered as a spin-off activity of the GIN group in order to feed back any experience of using early versions of open standards (e.g. BES, JSDL, SRM, GLUE2, UR, etc.) in Grid production setups to improve the standards wherever possible.

As a general paradigm, PGI focused first to agree on common terms that promote the mutual understanding. Based on this foundation use cases can be described that can be understood by the wide variety of PGI members belonging to different production Grids or technology providers. The use cases in turn provide the foundation for having a set of requirements that PGI aims to address in order to deliver additional profiles of available standards or improved standard specifications.

Until today, the PGI working group worked on a common glossary and surveyed various Grid uses cases common in production leading to distinct requirements in terms of standard specifications. The majorities of these use cases take lessons learned of production Grid infrastructures into account or raise the demand for specific functionality not covered yet by available open standards (i.e. BES, JSDL, etc.). While the glossary document is published separately, this particular document provides a survey of use cases common in the scientifically-driven e-science infrastructures and production Grids today. The contribution to the set of thirteen use cases is derived from members representing production Grids or specific relevant technology providers that we collectively name here as stakeholder of the aimed PGI set of specifications.

#	Use Case Name	Stakeholder
1	Virtual Physiological Human	IGE / Globus
2	Efficient MPI-based Parallel Job Executions	EGEE / EGI / gLite
3	Enforce Security of a Production Service Grid	EDGI
4	Prepare Desktop Grid Version of Application	EDGI
5	Marshal Activities from Service Grid to Desktop Grid	EDGI
6	High Capacity	GENESIS
7	Mid-Range Computing Use Case	GENESIS
8	Special Quality-of-Service Computing	GENESIS
9	GROMACS-based Molecular Dynamics in Bio-molecular Systems	GIN
10	Multi-Grid Drug Discovery Workflow	DEISA/UNICORE
11	RISM-FMO Coupled Simulation	NAREGI/RENKEI
12	Data intensive computation job processing on cluster-based Grids	NORDUGRID/ARC
13	European Grid Infrastructure	EGI

Table 1: List of stakeholders and use case names that collectively form the PGI use cases.

Table 1 clearly provides an overview of the impact of a potential PGI set of profiles or specification. On the one hand several important production Grid infrastructures participate regularly in the PGI working group sessions such as EGEE (more recently EGI), DEISA (potentially joined with PRACE soon), NAREGI/RENKEI, and NorduGrid. On the other hand, important technology providers in the field contribute with their experience and lessons learned in standard adoption and technology expertise. These are Globus, gLite, GENESIS, ARC, and UNICORE. In addition, the efforts in the PGI group are supported by the European Desktop Grid Initiative (EDGI) representing a stakeholder in the distinct area of desktop Grid infrastructures that in turn can be considered as a complementary infrastructure to scientifically-driven production Grids. Also the European Middleware Initiative (EMI) is supporting work carried out by members of the gLite, ARC, and UNICORE technology consortia. Finally, work in chairing the PGI efforts is also supported by the Standards and Interoperability for eInfrastructure Implementation Initiative (SIENA).

2. Use Cases Survey Structure

The structure of the use cases is largely based on the OGF OGSA-WG use case template (2nd revision). Nevertheless, we applied some minor changes to reflect the production Grid infrastructure context. This section shortly describes the structure for use case contributions that has been used in surveys.

While there are a lot of other structures that might be used the PGI working group agreed to use this structure initially to gather the use cases in a straightforward common manner.

2.1 Summary

This paragraph is used to provide context information about the use case and an overall high level summary description.

2.2 Customers

Use cases typically imply some form of customers that in this particular document might be e-scientists and/or computer scientists in the majority of the cases. Therefore, this paragraph provides a short abstract scenario description to explain customers' needs.

Questions relevant to this paragraph are: Where and how the use case occurs "in nature" and for whom it occurs? Is the use case inter-site or intra-site? Is it geographically distributed? How many users are expected for this use case?

2.3 Scenarios and Applications

This paragraph is used to explain the primary scenario of use cases and the applications used. One use case can stand for one or many scenarios that all are listed here.

2.4 Involved Resources and Production Grid Infrastructure

Under this heading all resources managed and provided by the Grid technology or infrastructure in context of the use case are explained.

Questions relevant to this paragraph are: Which infrastructure provided the resources? What types of resources are provided (HPC, HTC, both, etc.)? E.g. what hardware, data, software might be involved. Are these resources geographically distributed?

2.5 Security Considerations

This paragraph is needed to provide pieces of information about the required security environment as well as stating possible threats of the use case. This also includes, for instance, technical challenges by explaining why the delegation of rights is needed.

2.6 Performance Considerations

Performance considerations of the use case are explained in this paragraph. One question might be whether the performance of the submission is more relevant than the performance of the execution itself and so forth.

2.7 Use Case Situation Analysis and PGI Expectations

This paragraph discusses what services are relevant to the use case are already in production and to what extent they are satisfactory/unsatisfactory. This description may also include an articulation of what else needs to be done in tuning specific standards by supporting distinct missing concepts.

3. Virtual Physiological Human Use case

3.1 Summary

The Virtual Physiological Human (VPH) project is conducting research to provide computer models for personalized patient specific healthcare and also trying to create tools for modelling and simulation of human physiology (see [1]).

The scientists participating in the process are performing molecular dynamics simulations which are mostly non-interactive, although some small interaction is necessary. The users want to execute applications that use the SAGA API in order to access the resources and orchestrate the workflow. A SAGA compliant file transfer mechanism (e.g. GridFTP) should be used for the necessary file exchange and a job execution service with which SAGA could communicate (in this case Globus 5.0) is urgently needed.

3.2 Customers

For this use-case the customers are scientists participating in the VPH Virtual Community, working on molecular dynamics simulations. They would like to use Teragrid and DEISA resources for their computations. Currently the resource providers include the European HPC centres EPCC, SARA, LRZ and also TeraGrid resources within the US. The process can be separated into two steps. The first consists of simulations of so-called “ensembles” which are non-interactive and thus their computation can be considered embarrassingly parallel. The second step consists of loose coupling of the simulated ensembles, which requires limited interaction between the jobs.

3.3 Scenarios and Applications

As mentioned above, there are two steps, which can be considered as two different scenarios. The target is the simulation of a large number of ensembles. They are using SAGA for the workflow with Globus 5.0 at the time of writing. Although the WS stack is not needed since SAGA for Globus can only speak to the pre-WS stack, Globus will in the future provide CRUX as the WebService interface. That in turn will enable interoperability to other middleware. In addition, the SAGA interfaces could also provide an interoperability layer.

1) The first step consists of ensemble simulations, which is non- interactive. The user authenticates and submits a program called “replica exchange manager” along with the necessary information at the beginning. The replica exchange manager is responsible for initiating jobs to simulate ensembles. The simulations can last several hours on several hundred CPUs, but there is a host of ensembles that need to be simulated.

2) ‘Replica exchange’. In the second step, a loose coupling of the simulated ensembles takes place. Assuming that there are several MD simulations running they would need some interaction, for which they are using an advertising service based on SAGA. The interaction would not exceed the data size of several thousand bytes at about every 10000 steps. The data transfer is handled by the exchange manager via file exchange and according to this exchange some simulations are abandoned and others start. In order to avoid waiting in the queue each time a simulation is abandoned, they are using pilot[1] jobs that have been submitted and start actually processing after they receive the information to start the new simulation.

[1] Pilot jobs do not adhere to the SPG ‘Policy. In this context they refer to a single-user mode job that waits for input until it submits the actual activity.

3.4 Involved Resources and Production Grid Infrastructure

Involved resources include HPC resources from the production Grid infrastructures TeraGrid and DEISA.

DEISA Resources in particular are as follows:

- EPCC: Hector. 44.544 cores, 59.4Tb Mem, Peak 360TFlops
- SARA: Huygens. 3328 cores, 15.25Tb, Peak 60Tflops
- LRZ: HLRB2. 9728 cores, 39Tb, Peak 62.3Tflops

The resources provide file exchange and job submission interfaces which can be accessed by SAGA.

3.5 Security Considerations

The resource providers grant access, to users who authenticate using certificates or proxies (this can also be done through programs using the SAGA API). Another security consideration concerns the data exchange between instances running on different resources.[2] The programs should be able to communicate credentials across those resources to allow the data exchange to take place. Delegation is done via proxies for the file transfers both internally within a resource but also externally by the manager instances.

3.6 Performance Considerations

The use of HPC resources is of course advocated since there are hosts of replicas to be computed and the internal communication is limited. They are focusing on optimal use of the resources they have access to and this is the reason for the use of pilot jobs. The computations are highly scalable on the job level (NAMD and through it MPI are used), but also geographically, as resources from a variety of resource providers can be used.

3.7 Use Case Situation Analysis and PGI Expectations

The use case has the following expectations in terms of a PGI specification/profile:

- Storage resources for the interaction of the jobs are necessary and should be advertised.
- Furthermore NAMD is needed for the simulations. It would be, thus, nice to be able to ask an information service regarding software availability.
- File transfer services such as GridFTP.
- Advertising of services and endpoints is also necessary.
- The ability to submit multiprocessor jobs (MPI) is also needed.
- Authentication and authorization services for user authentication and file transfers

[2] The ability to reach the advertising service provided by SAGA as a separate program, is a concern but not in this context.

4. Efficient MPI-based Parallel Job Executions

4.1 Summary

The Grid community runs every day applications from research domains as diverse as multimedia, finance, archeology, and civil protection, astronomy, astrophysics, computational chemistry, earth sciences, fusion, and computer science. Often these applications solve complicated problems by implementing parallel algorithms which require adequate parallel computing mechanisms for their execution in order to enable the efficient usage of high-end large clusters or supercomputers.

The job classification refers as parallel a job composed by a large collection of tasks running on different nodes on the same site. Moreover inter-tasks communication is basically performed by using parallel computing mechanisms as the *Message Passing Interface (MPI)*. Moreover, it is consolidating the use of multi-threaded or Shared-Memory-device MPI codes on SMP (Symmetric Multi-Processing) Grid resources (that are becoming increasingly common).

The use case of parallel jobs in this Section is often used in the context of HPC-driven e-science infrastructures.

4.2 Customers

Customers belong to different research domains as multimedia, finance, archeology, civil protection, astronomy, astrophysics, computational chemistry, earth sciences, fusion, and computer science.

In general, customer can be seen as end-users of resources that enable the usage of the High Performance Computing (HPC) paradigm with massively parallel jobs.

4.3 Scenarios and Applications

This use case is valid for a wide variety of different massively parallel applications. We give three examples that take advantage of parallel jobs.

GARANT is an algorithm for automatic resonance assignment using as input the primary structure of a protein and lists with observed cross peaks from various spectra; knowledge about magnetization transfer pathways in the NMR experiments used is read from a library. The basic concept of GARANT is the matching of observed cross peaks to expected cross peaks derived from the sequence and the magnetization transfer pathways.

MOOSE (Multi-scale Object-Oriented Simulation Environment) is a general purpose simulator for biological systems. It is the base and numerical core for large, detailed simulations including Computational Neuroscience and Systems Biology. MOOSE spans the range from single molecules to sub-cellular networks, from single cells to neuronal networks, and to still larger systems. MOOSE is open source software, licensed under the LGPL (Lesser GNU Public License).

BEM (Biased Exchange Metadynamics) is an algorithm which allows reconstructing the free energy surface of complex systems (e.g. biomolecules in an explicit solvent), as a function of a large number of reaction coordinates: several replicas of the system are evolved in parallel at the same temperature by molecular dynamics, biasing a different reaction coordinate in each replica, and allowing the replicas to exchange the biasing potential from time to time. Implemented within the MPI scheme in a modified version of the GROMACS code, this technique has proved successful in calculating the detailed free energy surface (in explicit solvent) of both the 20-residues Trp cage and 36-residues Advillin headpiece proteins, in excellent agreement with experiments.

4.4 Involved Resources and Production Grid Infrastructure

The hardware resources involved are heterogeneous and geographically distributed. Some of them, as an example, are based on linux PCs, x64 architecture, >2GB RAM, single/multi core and/or single/multi processor.

The EGEE infrastructure provided these resources as the DEISA infrastructure does.

Production Grid services in this context are services in the context of computing elements (CEs), storage elements (SEs), as well as meta-schedulers or brokers such as the gLite Workload Management Systems (WMS).

4.5 Security Considerations

The following security aspects are essentials:

- User authentication and authorization mechanisms based on X509 certificates and VOMS attributes
- Delegation mechanism for allowing services (e.g .CE, WMS) to execute data-management operations in behalf of the user;
- Accounting system for charging the resource consumption;
- Encryption/protection of data on grid storage elements
- Data confidentiality, integrity, availability, authenticity, non-repudiation must be guarantee.

4.6 Performance Considerations

Typically the MPI applications are CPU and I/O intensive and so the performance aspect is essential in the underlying hardware resource. The Grid service performance providing access to those systems, on the other hand, is not that much important.

4.7 Use Case Situation Analysis and PGI Expectations

The use case has the following expectations in terms of a PGI specification/profile:

Although the OGSA-BES and JSDL specifications provide functionalities for creating and managing jobs on remote computational resources, they don't provide full support for job parallelism activities. A lot of profiles are available to fill the gap but they are often a complex mechanism that does not encourage the user to use it. Also, these profiles or standard extensions do not allow using the massively parallel systems in a very efficient manner that could be done by better specifying hardware setups (e.g. network topologies).

A better solution would be to extend the OGSA-BES and JSDL specifications by integrating the profiled items or, create a new specification which covers all these aspects.

5. Enforce Security of a Production Service Grid

5.1 Summary

This Use Case presents the *generic Security setup and process* which are necessary to securely operate a *Production Service Grid*. It stands as precondition for all other Use Cases in Production Service Grids.

A Production Service Grid must enforce *precise authentication of all Grid Users*, and traceability of their actions. This permits that on detection of any forbidden action, a *predefined procedure quickly transmits an alert containing precise information* to the Security Officers. *This precise information permits Security Officers to quickly assess the threat and undertake protective actions*.

5.2 Customers

This is *not* a commercial Use Case, so there are *no* customers. But there are Actors:

- *Trust Anchor*
  For example IGTF (International Grid Trust Federation)
- *Security Engineer of a Site* (of the System)
- *User Domain Manager*
- Grants access rights to the User
  The concept of User Domain is defined by GLUE 2.0
  An example of User Domain is a VO (Virtual Organization)
- *User (of the Service Grid)*
  Member of a User Domain
- *RA (Registration Authority)* having vetted the User
  For example an accredited manager of the Organization employing the User
- *CA (Certificate Authority)* having signed the credentials of the User
  For example GRID2-FR
- *CSIRT (Computer Security Incident Response Team)* of the System
  For example OSCT (Operational Security Coordination Team) was EGEE's CSIRT

5.3 Scenarios and Applications

This Use Case is *not* specific to particular applications. The scenario is as follows:

Preconditions

- The Production Service Grid is *setup and working* in operation with its Trust Anchor, its Operational Teams, its secure communication facilities
- The Trust Anchor has used a secure functionality of the System (for example the EUGridPMA repository) to *publish an updated list of Certificates* and CRLs (Certificate Revocation Lists) of the CAs accredited on the System.
- Each Security Engineer has *deployed this updated list of CA Certificates* on all resources under his/her responsibility (which belong to the System).
- A User, authenticated by a Certificate issued by a CA upon vetting by an RA, and belonging to an UserDomain, has used the System to *obtain Security Credentials* (for example RFC-compliant X509 proxy, Bag of SAML assertions).
- The above mentioned User has employed these Security Credentials in order to *make use of a resource of the System* (for example he/she accesses Data stored inside the System, submits an Activity[3] to the System, ...).

Triggers

After a certain time, the System *sends a Security Alarm* to a Security Engineer on duty for a Site of the System.

[3] 'Activity' is defined by GLUE 2.0 as 'Unit of work which is submitted to a Service via an Endpoint'

Basic course of events

1) Upon reception of this Security Alarm, this Security Engineer *investigates the Security Records* of the System (for example Activity Logs) and obtains first proofs that the above User has tried to perform forbidden actions (for example Installation of a root kit, Mass mailing, ...).

2) This Security Engineer immediately :
   - *freezes* all tasks created locally for this malicious User, and all incoming requests of this malicious User,
   - uses a secure functionality of the System to *send a Security Alert* containing these first proofs to the CSIRT.

3) Upon reception of these first proofs, the CSIRT examines them to *verify that this User is malicious* indeed, and extracts from his/her Security Credentials :
   - The identity of the UserDomain having issued access rights to this User,
   - The identity of the CA having signed his/her Certificate.

4) The CSIRT immediately uses secure functionalities of the System to :
   - *publish a Global Ban* of this User from the System,
   - *send a confirmed Security Alert* to the Users's UserDomain Manager, and to the above identified CA.

5) Upon reception of this Global Ban, *each Security Engineer locally bans* this User from all resources under his/her responsibility.

6) Upon reception of this confirmed Security Alert :
   - the UserDomain Manager *suspends or revokes membership* of this User,
   - if necessary, the CA *adds* the Certificate of this User to its *CRL*,
   - the CA uses secure functionalities of the System to *forward this confirmed Security Alert* to the RA having vetted this User.

Postconditions

- *Persistent logs* of following actions are available for further security risk assessment, legal matters, … :
  - actions of the malicious User,
  - communications between persons responsible for Security.
- All processes of the malicious User are *frozen*, which permits generation of useful system dumps and detailed post-mortem analysis.
- The malicious User is globally *banned*, and can *not* perform any action on any local resource of the Service Grid.
- The malicious User is *not* member of the UserDomain anymore, and can *not* perform any action on behalf of this UserDomain anymore.
- The Certificate of the malicious User is *not* valid anymore. That *forbids* him to use it successfully anymore.
- The *RA* having vetted the malicious User has received a confirmed Security Alert, and can take disciplinary actions as needed.

Alternative paths or Extensions

- On reception of the Security Credentials of the User, the System immediately detects that they are *not* valid (for example match with CRL) and rejects the Request of the User without any further processing.
- The Security Engineer receiving the Security Alert is *not* trained enough to investigate the Security Records.
- Security Records contain only local identifiers, but *no* reference to the Security Credentials of the User, so that Security Engineers cann*ot* identify the malicious User.
- Instead of freezing all tasks created locally for this malicious User, the Security Engineer *destroys* them, which *forbids* later generation of useful system dumps permitting detailed post-mortem analysis.
- The Security Engineer having identified the malicious User does *not* send a Security Alert to the CSIRT.
- The Security Engineer having identified the malicious User bans the malicious User locally *before* receiving the Global Ban of this User, taking the risk that the system automatically reroutes attacks of this User to other Sites of the System, which are still vulnerable.

5.4 Involved Resources and Production Grid Infrastructure

The System is the *whole Production Service Grid*, including its Security Infrastructure, its Operational Teams, its secure communication facilities, and all Data and Computing Resources which can be used directly or indirectly (taking into account possible migration of Activity). Examples are EGI, EELA, and DEISA.

The Data and Computing Resources are *not* managed centrally inside a unique administrative domain, but locally on Sites. Each Site is managed locally as a separate administrative domain.

The OO principle of encapsulation is enforced by the Grid middleware which hides the site details for standard usage of Security Credentials and processing of Activities. But this principle of encapsulation cann*ot* be enforced for Security alerts, which are detected and processed first in a specific administrative domain.

5.5 Security Considerations

Security is the very focus of this Use Case.

A Production Service Grid must:

- adequately *train* the persons responsible for Security,
- regularly *perform global Security Challenges* and audit the results.

A Production service Grid cann*ot* operate securely if it does not implement a Use Case such as this one.

5.6 Performance Considerations

For this Use Case, performance is a tradeoff between user friendliness and achieved security.

In particular, for X509 certificates, the process of securely transferring a PKCS12 bundle from the browser of a user to separate ‘userkey.pem’ (access rights = 400) and ‘usercert.pem’ (access rights = 444) files inside the ‘~/.globus’ folder of this user is very tedious. A tool automating this process would be welcome.

5.7 Use Case Situation Analysis and PGI Expectations

For Grid interoperability, this Use Case triggers following requirements:

– Robust standardized *Information System* describing Grid entities to be monitored for security. The relevant standard is OGF GLUE 2.0 (GFD.147) [4] which is currently being implemented by major Grid infrastructures.

– Robust standardized *Security* design: Credentials definition, Setup, Process, ... Relevant official standards are X509 (RFC 2459), SSL / TLS (RFC 2246/2818), X509 Proxy (RFC 3820), SAML (Oasis), COPS (RFC 2748), and perhaps the standards based on WS-Security (Oasis). Relevant de facto standards are IGTF and VOMS (used by EGI).

– Robust standardized *Monitoring* permitting to quickly detect security issues. A standard, such as proposed by OGF OCCI Req. Table 7, would be useful.

– Robust standardized *Logging* and *Accounting* holding *persistent* records permitting easy Security audits. An existing implementation is Logging and Bookkeeping (gLite). Activity Instance Documents Specification (currently published for public comment by OGF) is perhaps relevant.

6. Prepare Desktop Grid Version of Application

6.1 Summary

Service Grids (Federations of managed computing resources) trust Users and *push* Activities[4], while *Desktop Grids* (Loose opportunistic Grids using idle resources) trust Applications and *pull* Activities. So, there is *no* interoperability at all. Interoperation can be achieved only by using a *Bridge* specially designed for this purpose.

This Use Case presents the *preparatory steps* which are necessary to *permit submission of an Application on a Service Grid for execution on a Desktop Grid through an adequate Bridge*. This goal is achieved more easily if the Application already runs on the Service Grid, but this is *not* an absolute prerequisite (an example is an MS-Windows version of an Application).

The Bridge does *not* simply forward the Activities from one Grid to another Grid, but it is a stateful server performing extensive work summarized here as '*marshaling*'. Detailed description of this Bridge is outside the scope of the Use Case : Only the Bridge Manager needs to have a detailed knowledge of it. The other Actors only need to know that this Bridge exists, and trust it.

- A suitable *Application* is chosen, and a *Desktop Grid version* of it is developed, tested, validated, and stored in an *Application Repository* by qualified and authorized IT professionals.
- A *UserDomain Manager* creates *specific Credentials* to identify Activities suitable for submission to a Service Grid and execution on a Desktop Grid, and grants them to *suitable members* of the UserDomain.
- The *Bridge Manager configures the Bridge* to associate a Bridge Endpoint to these specific Credentials, and publishes this association.

6.2 Customers

This is *not* a commercial Use Case, so there are *no* customer.

But there are Actors, who all are qualified and authorized IT professionals (in opposition to Application Users and Scientists having no advanced IT training):

- *Application Gridification Team*
  For example the team having already gridified the Application for a Service Grid
  - *Gridification* means the complete software lifecycle necessary for an Application to correctly run on a Grid.
  - Gridification of the Application for a *Service Grid* is a completely separate Use Case (*not* described here).
- *DG Application Validation Team*
  Team Validating that the Desktop Grid version of the Application is really suitable for submission to the Desktop Grid through the Bridge
  An example of such team is EDGI SA2
- *DG Application Repository Manager*
  Manager of the Application Repository for Desktop Grid Applications
  An example of such Manager is a member of EDGI SA1
- *UserDomain Manager*
  Grants access rights to the User
  The concept of UserDomain is defined by GLUE 2.0
  An example of UserDomain is a VO (Virtual Organization)
- *Bridge Manager*
  Manager of the Bridge marshaling the Activities from the Service Grid to the Desktop Grid
  An example of such Manager is a member of EDGI SA1

[4] 'Activity' is defined by GLUE 2.0 as 'Unit of work which is submitted to a Service via an Endpoint'

6.3 Scenarios and Applications

Applications have to be suitable for execution inside a Desktop Grid :

- *Input and Output files well known* at the time of submission of the Activity,
- *no* Application License,
- *no* Manual Data Staging,
- *no* Hold Point.

The *Scenario* is the following :

Preconditions

The *Security Context* is described by the EDGI Use Case 'Enforce Security of a Production Service Grid'.

Triggers

An Application Gridification Team, belonging to a UserDomain, chooses an *Application* which is suitable for execution inside a Desktop Grid (see above).

Basic course of events

Following steps must be performed only once for each (Application, Desktop Grid) pair.

1) The Application Gridification Team :
   - develops the Desktop Grid Version of the Application,
   - tests this Desktop Grid Version on a Test Infrastructure,
   - *delivers this Desktop Grid Version* to the DG Application Validation Team.
2) The DG Application Validation Team :
   - uses a Validation Infrastructure to verify that the Desktop Grid version of the Application can really be submitted through the Bridge for successful execution on the targeted Desktop Grid, and is harmless to the resources of the Desktop Grid,
   - *issues an official document stating that this Application version is suitable indeed*, and transmits this document to the DG Application Repository Manager.
   - *certifies this Desktop Grid Version for DG use* and notifies the DG Application Repository Manager.
3) The DG Application Repository Manager :
   - *stores the validated Desktop Grid Version* inside the Application Repository of the Operational Bridge,
   - notifies the UserDomain Manager to which the Application Gridification Team belongs.
4) The UserDomain Manager :
   - *creates Credentials* (such as a VO Group or a VO Role) permitting to identify Activities suitable for submission to one Desktop Grid through the Bridge, and informs the Bridge Manager,
   - *grants these specific Credentials* to suitable members of the UserDomain, and informs them.
5) The Bridge Manager :
   - *configures the Bridge* to associate a Bridge Endpoint to these specific Credentials,
   - *publishes this association* to the Information Repository of the Service Grid.

Postconditions

- The validated *Desktop Grid Version of the Application* is stored inside the *Application Repository* of the Bridge.
- *Specific Credentials* are created inside the *UserDomain server* for identification of Activities suitable for submission to one Desktop Grid through the Bridge.
- A *Bridge Endpoint is associated to these credentials*, and this association is published to the *Information Repository* of the Service Grid.

Alternative paths or Extensions

The Desktop Grid version of the Application :

- *crashes* on the Validation Desktop Grid, and is NOT validated,
- *overflows* the Bridge resources of the Validation Infrastructure, and is NOT validated.

6.4 Involved Resources and Production Grid Infrastructure

The System is a *Production Service Grid* equipped with a *Bridge marshaling Activities to Production Desktop Grids*.

The system fully includes the Production Service Grid, the Bridge, and all connected Desktop Grids.

Examples are EGI, EELA, DEISA using the EDGI 3G Bridge to submit Activities to SZTAKI DG, Ibercivis, AlmereGrid, OurGrid.

The goal is to enforce the OO principle of encapsulation as much as possible for the *Users* who will submit Activities for execution on the Desktop Grid.

Since Desktop Grids accept only *validated Applications*, this encapsulation goal requires *IT professionals* to perform the current Use Case on *appropriate specific internal components* of the System.

6.5 Security Considerations

The *Security Context* is described by the EDGI Use Case 'Enforce Security of a Production Service Grid'.

6.6 Performance Considerations

For this Use Case, performance is a tradeoff between the complexity of Application gridification and the huge computing power available for the Application once it has been gridified.

6.7 Use Case Situation Analysis and PGI Expectations

For Grid interoperability, this Use Case triggers following requirements:

- Standardized *Information System* describing Grid entities able to accepts Activities. The relevant standard is OGF GLUE 2.0 (GFD.147) [4] which is currently being implemented by major Grid infrastructures.
- Standardized *Security* design permitting to associate specific Resources to Credentials. The relevant de facto standard is VOMS (used by EGI).
- Standardized *Application Repository*. We do *not* know of any relevant standard.

7. Marshal Activities from Service Grid to Desktop Grid

7.1 Summary

Service Grids (Federations of managed computing resources) trust Users and *push* Activities[5], while *Desktop Grids* (Loose opportunistic Grids using idle resources) trust Applications and *pull* Activities. So, there is *no* interoperability at all. Interoperation can be achieved only by using a *Bridge* specially designed for this purpose.

This Use Case presents the *processing of an Activity submitted on a Service Grid for execution on a Desktop Grid*. Detailed description of the Bridge marshaling Activities from Service Grids to Desktop Grids is outside the scope of this Use Case: The User wishing to submit an Activity only needs to know that this Bridge exists, and trust it. Other Actors of this Use Case do *not* even need to know that this Bridge exists.

- IT professionals must already have performed the '*Prepare Desktop Grid Version of Application*' Use Case for the Application to be executed within the Activity.
- A User obtains appropriate Security Credentials and *submits an Activity* referencing the validated Desktop Grid version of the Application.
- The System recognizes these specific Security Credentials, and brokers the Activity to the associated *Bridge, which marshals the Activity* and any subsequent User request to the Desktop Grid.

7.2 Customers

This is *not* a commercial Use Case, so there are *no* customers. But there are Actors:

- *A User (of the Service Grid)* wishing to submit an Activity
  Member of a User Domain
- *Any User (of the Service Grid)*
  Member of a User Domain

7.3 Scenarios and Applications

Applications have to be suitable for execution inside a Desktop Grid :

- *Input and Output files well known* at the time of submission of the Activity,
- *no* Application License,
- *no* Manual Data Staging,
- *no* Hold Point.

The *Scenario* is the following:

Preconditions

- The *Security Context* is described by the EDGI Use Case 'Enforce Security of a Production Service Grid'.
- IT professionals must already have performed the '*Prepare Desktop Grid Version of Application*' Use Case for the Application to be executed within the Activity.

Triggers

A User, member of a User Domain, decides that an Activity has to be submitted on the Service Grid for execution on a Desktop Grid.

Basic course of events

1) The User makes usage of the System to *obtain Security Credentials* which are appropriate for the validated Desktop Grid version of the Application to be executed within the Activity.
2) Optionally, this User fetches the validated Desktop Grid version of the Application.
3) This User employs the Security Credentials obtained above to *submit to the System an Activity*:
   - containing the *validated Desktop Grid version of the Application*, or referring to it,

[5] 'Activity' is defined by GLUE 2.0 as 'Unit of work which is submitted to a Service via an Endpoint'

- containing Input Data not too large, or referring to Input Data of any size already present in a Storage Resource readable by the Bridge with the User Credentials.

4) The System *performs its usual processing*: It *looks up its Information Repository with the User Credentials* and *brokers the Activity* to the associated Endpoint, which, if all goes well, is the appropriate Bridge Endpoint.

5) The Bridge *processes the Activity* following the usual protocol of the Service Grid. In particular:
   - as soon as possible, the Bridge *returns to the User an Activity ID* permitting easy traceability of the Activity,
   - each time any User *issues a subsequent request* on this Activity, (Status query, Output retrieval), the Bridge Endpoint *responds* with its best knowledge if the User is authorized, otherwise it rejects his/her request.

Postconditions

The System keeps persistent detailed Log and Accounting for the Activity, so that any authorized User can query them, as well during the lifetime of the Activity as after the Activity has finished.

Alternative paths or Extensions

- The *User is not trained enough* to obtain the required specific Credentials, so that the System does *not* broker the submitted Activities to the Bridge.
- The System does *not correctly use its Information Repository*, and does *not* broker the submitted Activities to the Bridge.

7.4 Involved Resources and Production Grid Infrastructure

The System is a *Production Service Grid* equipped with a *Bridge marshaling Activities to Production Desktop Grids*.

The system fully includes the Production Service Grid, the Bridge and all connected Desktop Grids.

Examples are EGI, EELA, and DEISA using the EDGI 3G Bridge to submit Activities to SZTAKI DG, Ibercivis, AlmereGrid, OurGrid.

- The goal is to enforce the OO principle of encapsulation as much as possible for the *User* who submits Activities on the Service Grid for execution on the Desktop Grid.
- The User still has to:
  - assess if the Input Data to be processed and the Output Data to be obtained can be made publicly available or not. If *not*, this User should submit the corresponding Activity for execution only to the local Desktop Grid of his/her Institution, *not* a public Desktop Grid.
  - obtain the appropriate Credentials for the validated Desktop Grid version of the Application.
- Once the User has retrieved the appropriate Credentials, the system fully enforces the principle of encapsulation. The Client interface stays unchanged, and *the Bridge stays the unique Endpoint for User requests for the whole life time of the Activity*: When necessary, the Bridge marshals User requests to the Desktop Grid, and marshals back Desktop Grid Responses to the User.

7.5 Security Considerations

The *Security Context* is described by the EDGI Use Case 'Enforce Security of a Production Service Grid'.

7.6 Performance Considerations

Once the Application has been gridified (adapted for submission) on Desktop Grids, usage of the Bridge to Desktop Grids is a huge performance gain for the User.

7.7 Use Case Situation Analysis and PGI Expectations

For Grid interoperability, this Use Case triggers following requirements:

- Robust standardized *Information System* describing Grid entities able to accepts Activities. The relevant standard is OGF GLUE 2.0 (GFD.147) [4] which is currently being implemented by major Grid infrastructures.
- Robust standardized *Security* design permitting to associate specific Resources to Credentials. The relevant de facto standard is VOMS (used by EGI).
- Robust standardized *Application Repository* permitting Users to pick an Application version which is adequate for execution inside a Desktop Grid. We do *not* know of any relevant standard.
- Robust standardized *Logging* holding *persistent* records permitting Users to audit job history (for example in case of error). An existing implementation is Logging and Bookkeeping (gLite). Activity Instance Documents Specification (currently published for public comment by OGF) is perhaps relevant.
- Robust standardized *Accounting* holding *persistent* records permitting Users to analyze resource usage. The relevant standard is OGF UR (GFD.98). We do *not* know if current implementations really use it and are interoperable.
- Standardized *Job Description Language* permitting Users to describe Input Data, Output data and other requirements in conformance with GLUE 2.0.

8. High Throughput or High Capacity Computing

8.1 Summary

High-throughput computing (HTC) refers to running many independent instances of the same job at the same time. The different instances may be the same problem with different parameters - as in a parameter space study; with different input files - as in a comparison of a 3D structure against many other known structures; with different frame numbers - as in the generation of a movie; with many different seeds - as in a *Monte Carlo* simulation. The job instances may be sequential (i.e., a typical parameter sweep or HTC use) or parallel (a common HPC use).

Typically, the client launches the job set over a short period of time and then monitors progress, usually via a script, a workflow engine such as DAGMAN, or a science gateway. The locus of control of the workflow engine or script is often a front-end node at a computing center, or a community scientific gateway/portal or an end- user machine.

Because the job(s) data may not be located where execution will occur, the data may need to be copied. Access to the data may be provided via one of many different data movement protocols, including a wide-area or federated file system, http, ftp, sftp, gridftp, SRM, scp, iRODS, or some other tool. Oftentimes one or more of the input files to the computation is “constant”, i.e., it is invariant across a large number of runs. This can facilitate simple optimizations that only copy the file once, rather than once for each job instance.

An executable may come in many forms. It may be a binary generated from source code under the users’ control, a package with no source, a shell script, or a program in an interpreted language such as Matlab, R, Python, or Perl. In addition to the executable itself, the application may require particular versions of particular libraries. In addition to the management of different versions of different executables and libraries, licensing issues may also be a concern.

8.2 Customers

High-throughput computing is used by many different disciplines: physics (HEP particularly), chemistry, computer science, computer engineering, systems engineering, biochemistry, biology, economics, public health science, and chemical engineering—just to name a few.

8.3 Scenarios and Applications

Sample scenario 1:

The Southern California Earthquake Center (SCEC) [5] is an inter-disciplinary research group with over 600 geoscientists, computational scientists and computer scientists from ~60 institutions including the US Geological Survey. Its goal is to develop an understanding of earthquakes, and to mitigate risks of loss of life and property damage. An important aspect of SCEC research is in the CyberShake project, which calculates probabilistic seismic hazard (PSHA) curves for sites in Southern California. A PSHA map provides estimates of the probability that the ground motion at a site will exceed some intensity measure over a given time period. For each geographical point of interest, two large-scale MPI calculations and approximately 840,000 data-intensive post processing jobs are required. That is, the 100-200 TB datasets generated by the large-scale runs must remain available for the entire time period that it takes to run 840,000 serial jobs.

Sample scenario 2:

Computational crystal structure prediction (CSP) typically requires the processing of an enormous workload in the form of thousands of small jobs. Professor Sally Price's research group at UCL is developing the accurate modeling of intermolecular and intramolecular forces, in order to predict which crystal structures of an organic molecule are thermodynamically feasible. These are contrasted with experimental searches for polymorphs in order to understand the factors which lead to polymorphism, in a multi-disciplinary project "Control and Prediction of the Organic Solid State" (CPOSS) [6].Crystal-structure prediction answers the question 'what crystal structures will an organic molecule adopt?' Different researchers have united to break that question down into two smaller steps 'what crystal structures could this organic molecule adopt?', and 'which structure is the molecule most likely to adopt?' These two questions naturally separate the computational procedure into two sequential steps:

- The generation of a bank of structures that the molecule could adopt (usually based on very simple-to-evaluate criteria).
- The calculation of lattice energy for each structure, which can be used to estimate the structure's stability. Determining the lattice energy of a structure requires energy calculations for thousands, or tens of thousands, of possible crystal structures.

8.4 Involved Resources and Production Grid Infrastructure

HTC jobs may be run on a variety of resources - within a computing center site, on a campus grid, or on research clusters or across resources spanning multiple sites, grids or clusters.

Functional Requirements

Job Management: Basic job description and management capabilities are required to describe, schedule, start, monitor, interact with, and clean up jobs. Once initiated, clients—potentially including software clients such as workflow engines or science gateways—require a mechanism to uniquely name (identify) jobs for subsequent management and interaction.

Job Specification: The job description must contain information regarding the data sets, resource requirements, dependencies on other jobs, etc.

Scheduling: Scheduling the job consists of at least three logical phases: determining where the job can run based on resource and account requirements, selecting a resource on which to execute the job, and preparing the execution environment for executing the job (e.g., staging data, getting the binaries in place, etc.) Determining where the job may run may require an information system that describes execution environments, their capabilities, software installed, etc. The ability to start both single and "vector" batches of jobs should be supported.

Job Interaction and/or Steering: Interaction with a job that is in progress is desired. In particular it is desirable to be able to access and manipulate the files in the working directory of a running job – for example to steer the job, check its progress or debug it.

Job Status: Monitoring job status/progress via traditional client polling is necessary while support for some sort of notification is desirable but not necessary. A clearly described state model for execution must be supported. The ability for an end-user or automated tool to determine the root cause of job failure is also desirable. Some clients require that job data not be immediately reaped upon completion, therefore the grid infrastructure must support the ability for a job to specify whether to delay automatic clean up of job data.

8.5 Security Considerations

Authentication to execution is usually necessary. Similarly, if the execution service is to access data elsewhere via another service, that service may require authentication as well. This in turn may require some form of delegation from the client initiating the set of jobs to execution services that will move data in and out of the execution location.

Besides the authentication, access control, and delegation issues, some users have data integrity concerns with their data. This affects both on-the-wire and (more importantly) on-disk storage of input data and results.

In regard to groups and virtual organizations, access to compute resources and data resources may require one or more of the following: individual authentication, authentication as a member of a group or virtual organization, or the assertion of some property or role. Similarly, jobs may need to be managed post-initiation by either the identity that started the job or members of a group or role, or both.

8.6 Performance Considerations

Throughput (in jobs per second) is the primary performance metric for the HTC system itself. Staging time is also critical in terms of end-to-end execution time, as is task scheduling, in order to reduce the amount of staging necessary in the first place.

With respect to availability and reliability, users can accept that occasionally the HTC system is unavailable. Lost jobs or jobs that fail for no identifiable reason is not desirable and must be minimized.

See https://twiki.cern.ch/twiki/bin/view/EGEE/SA3Testing#CREAM_CE for sample performance, reliability, and availability metrics used in g-lite/EGEE.

8.7 Use Case Situation Analysis and PGI Expectations

This use case relies on the ability to schedule, start, stop, monitor and manage jobs of various types as well as staging data to and from jobs and accessing intermediate job data.

At the moment, the existing BES, JSDL, and JSDL extension specifications can be used to support most of functionality required for the execution of, and management of jobs as well as staging data to and from jobs.

One area not directly covered by the existing specifications, is the ability to access the intermediate data of a job in progress. However, there are several existing specifications that could be adopted to provide part of this functionality. In particular, the RNS specification can be used to model the directory structure of the job's working directory and provide an interface to access and manipulate directory entries; the ByteIO specification(s) can be used to model data files and provide the interface to manipulate them; and the BES specification could be extended to provide a mechanism to access the root of a job's working directory – for example, by profiling BES to define the interface for the BES Activity port type to include this functionality or by adding a new port type to the BES specification.

Additional profiling or extensions of JSDL and BES may be required for some functionality or to increase interoperability between grid implementations.

From a security perspective, existing OGSA security profiles and specifications cover many, but not all of the use case.

9. Mid-Range Computing Use case

9.1 Summary

Mid-range HPC computation is that which utilizes parallelism to solve larger problems than those which can be handled with desktop systems or campus-level clusters, but which do not utilize the large proportion (>25%) of capability HPC systems as described in the capability use case. This class of user is often in the early stages of scaling up their code from campus-level systems, or may be performing a data analysis task for which a large memory space and high-performance storage system is more important than an enormous number of processes. In addition, these jobs are often components of larger workflows, where the input to the job is the output of another task, or vice versa – this leads to an increased importance for data locality. The recent improvements in parallel and remote visualization techniques have led to the introduction of visualization users to this category as well.

Since these jobs tend to fit on a larger proportion of the systems in national cyberinfrastructures, and are more likely to be part of a workflow, meta-scheduling and co-scheduling are more likely to be of value to these users. In addition, the vast increase in available cores at the national scale has led to the introduction of “parameter sweeps” for mid-sized parallel jobs, where a whole set of problems with varying parameters may have computational demands which are high enough to require parallel and distributed processing. For this reason, there may be a need to launch many mid-sized jobs on one or more resources simultaneously or sequentially.

9.2 Customers

As with other HPC and HTC use cases, this class of jobs has a wide range of users from many disciplines. The fact that this class of user is often relatively new to national-scale cyberinfrastructure also means that newer disciplines which have not historically utilized HPC resources are disproportionately represented in the mid-range HPC category. User HPC sophistication can range from low to high, and users may be comfortable with interfaces from web portals and gateways to shells and text-based batch job construction.

9.3 Scenarios and Applications

The Network for Earthquake Engineering Simulation (NEES) project is an NSF MRE (Microbiological Research Establishment) project that seeks to lessen the impact of earthquake and tsunami-related disasters by providing revolutionary capabilities for earthquake engineering research. A state-of-the-art network links world-class experimental facilities around the country, enabling researchers to collaborate on experiments, computational modeling, data analysis, and education. NEES currently has ~75 users across approximately 15 universities. These users use TeraGrid HPC and data resources for various structural engineering simulations using both commercial codes and research codes based on algorithms developed by academic researchers. Some of these simulations, especially those using commercial FEM codes such as Abaqus, Ansys, Fluent, and LS-Dyna, require moderately large shared memory nodes, such as the large memory nodes of Abe at the Pittsburg Supercomputing Center, but scale to only a few tens of processors using MPI. Large memory is needed because the whole mesh structure is required on each node due to the basic FEM algorithm used. Many of these codes have OpenMP parallelization, in addition to using MPI, and users mainly utilize shared memory nodes using OpenMP for pre/post processing. On the other hand, some academic codes, such as OpenSees which is tuned for specific material behavior, can scale well on many thousands of processors, including on Kraken at NICS, and Ranger at TACC. Due to the geographically distributed location of NEES researchers and experimental facilities, high bandwidth data transfer and data access are vital.

9.4 Involved Resources and Production Grid Infrastructure

This use case has a very broad set of potentially involved resources, including medium-sized clusters and smaller percentages of the largest capability systems, visualization clusters and other data-centric systems, high-performance parallel file systems and archives at one or more sites. In addition, users in this category may need to access a shared community data collection, both to retrieve input data sets before a job begins or to store their output with appropriate metadata and other provenance information once a job has completed. Some jobs may be able to run on any one of several X86-based Linux clusters in particular, while other jobs may have other specialized requirements such as the need for GPUs, shared-memory architectures, or specific data sets local to a computational resource.

The size of data associated with this use case may vary from very small, such as a set of parameters to be passed to the executable, or as large as some multiple of the in-memory size of the total task, for jobs that perform checkpointing.

This use case contains the following functional requirements.

Resource access: Users of mid-range computing applications typically submit jobs directly, may make use of workflow tools or may use gateways to submit jobs.

Authentication, authorization, and accounting (AAA): These users require standard Unix and batch AAA as implemented at individual resource provider sites and following the policies those sites adhere to.

Fault tolerance: Some mid-range computing applications have built-in periodic checkpointing features to guard against loss of work in the event of a system crash. Some mid-range jobs may employ workflow restart capabilities for fault tolerance.

Scheduling: Scheduling the job consists of at least three logical phases: determining where the job can run based on resource and account requirements, selecting a resource on which to execute the job, and preparing the execution environment for executing the job (e.g., staging data, getting the binaries in place, etc.) Determining where the job may run may require an information system that describes execution environments, their capabilities, software installed, etc. The ability to start both single and “vector” batches of jobs should be supported.

Advance scheduling: Some mid-range computing applications may require special scheduling capabilities including pre-stage and post-stage data job scheduling or sequential job dependencies.

Workflow: Some mid-range computing applications have workflow components. Most of these workflows are managed by the users using homegrown tools or special workflow tools.

Data storage: The data sets produced by mid-range computing applications can range from small to quite large. The storage and archival resources available must be both sufficiently large to store such data sets as well as sufficiently fast to handle them in a timely manner.

9.5 Security Considerations

Authentication to execution is usually necessary. Similarly, if the execution service is to access data elsewhere via another service, that service may require authentication as well. This in turn may require some form of delegation from the client initiating the set of jobs to execution services that will move data in and out of the execution location.

Besides the authentication, access control, and delegation issues, some users have data integrity concerns with their data. This affects both on-the-wire and (more importantly) on-disk storage of input data and results.

In regard to groups and virtual organizations, access to compute resources and data resources may require one or more of the following: individual authentication, authentication as a member of a group or virtual organization, or the assertion of some property or role. Similarly, jobs may need to be managed post-initiation by either the identity that started the job or members of a group or role, or both.

This use case may have a somewhat greater need for the ability to support campus-level authentication, as it is likely the primary use case representing beginning users.

9.6 Performance Considerations

Performance considerations generally are similar to those of the capability computing use case – i.e. parallel job execution time and I/O performance are likely to be the most important components of overall job performance. As this use case may include situations where tasks are interdependent, and is less likely to depend on special queueing arrangements such as those used to favor the largest jobs in a queue, the reliable performance of scheduling systems and/or metascheduling systems may be somewhat more important than in the capability use case.

9.7 Use Case Situation Analysis and PGI Expectations

This use case relies on the ability to create and manage jobs on remote resources. In this case, there seems to be very few functions that aren't handled by existing BES, JSDL, and JSDL related specifications. Some amount of profiling may be desirable to achieve/encourage interoperability such as specifying mandatory supported data staging protocols and/or data staging protocol discovery for BES implementations.

From a security perspective, existing OGSA security profiles and specifications cover most of the use case.

10. Special Quality-of-Service Computing

10.1 Summary

Special quality of service (QOS) high performance computing and data jobs refer to a class of applications that require use of a specific HPC resource at a specific time, within a specific timeframe range, on a periodic recurring basis or on an urgent basis. The QOS types include high priority, advanced reservations and urgent computing. High priority QOS has no definite start time and should start as soon as possible. Advanced Reservation QOS has a definite start time and a reservation on the required resources and may include a periodic reservation.

Urgent Computing QOS has an urgent requirement and needs specific resources as soon as possible (essentially immediately when the job(s) are ready). These applications need high priority or guaranteed access to the required resources or services within a specific timeframe due to deadlines, due to time dependent collection and/or processing requirements of data from specialized instruments, for time dependent simulations such as climate modeling or to satisfy other high priority time dependent requirements. Use of manual or automated advanced reservations is usually employed on the end resources to provide the capability for QOS jobs. These reservations are usually manual, but automated methods are wanted for some applications. Standing reservations are used for the periodic recurring QOS jobs. Application types for QOS jobs include HPC capability jobs, HPC mid range jobs, and HPC data intensive jobs.

10.2 Customers

Beneficiaries include users with deadlines for work (perhaps for a paper, a journal, or a conference deadline) or with periodic model/simulation frequencies based on periodic data collection or periodic deadlines. QOS users come from a variety of different disciplines, including, but not limited to, physics, aerospace and mechanical engineering, meteorology, and climatology. They can be expert users either who develop their own large scale applications, have extensive experience modifying community applications in an HPC environment, are associated and have experience with a portal or gateway, or are familiar with direct batch job submission.

10.3 Scenarios and Applications

On Sept. 7, 2008, six days before Hurricane Ike, the third most destructive hurricane in U.S. history, crashed into the Texas coast, NOAA and NSF urgently contacted TACC to help answer several vital questions using computational analysis [7]. The multi-tiered effort used global models with twice the resolution of the best operational simulations, regional models six times as high-resolution as those now used, and added, for the first time, Doppler radar data streamed directly from NOAA's planes to TACC Ranger. A partition of Ranger was cleared and dedicated to running ensembles of thirty 1.5 km resolution WRF model jobs to predict landfall. Also in dedicated mode on Ranger, the ADCIRC model was used to predict the storm surge, while another server at TACC was engaged in evacuation logistics coordination management.

10.4 Involved Resources and Production Grid Infrastructure

High priority or dedicated computational, data and archival resources available at a specific site.

The following functional requirements exist for this use case.

Scheduling: For computational resources, access to a high priority queue or high scheduling priority in the resource management (batch) system.

Advanced Reservations: Ability either manually or in an automated way to request and obtain an advanced reservation for computational resources. The reservation can either be a single advanced reservation or a recurring periodic reservation. For data or archival resources, this is usually a manually made advanced reservation of adequate storage space.

Urgent Computing: These high priority on-demand large-scale computations can't wait endlessly in a job queue for supercomputer resources to become available. These jobs must have access to resources on an as soon as possible or “now” basis. Service providers need to have the capability to schedule these jobs as a high priority or preempt (or checkpoint/restart if available) existing jobs to run these “now”.

10.5 Security Considerations

This use case typically requires normal user account access and local and remote batch queue access.

10.6 Performance Considerations

This use case requires high priority or reserved access to resources.

10.7 Use Case Situation Analysis and PGI Expectations

This use case relies heavily on the ability to make reservations of resources (be they compute, or data), and at the proper time, execute jobs on those resources. Further, for the urgent computing aspect of this use case, the ability to manage jobs already running (either to kill, suspend, checkpoint, etc.) is needed.

At the moment, the existing BES, JSDL, and JSDL extension specifications can be used to support the execution of, and management of jobs. Since most of the functionality for execution in this scenario is implementation and not interface (i.e., the interface to BES doesn’t have to change to take into account urgent computing needs), few if any extensions are necessary. However, extensions would be needed to support the reservation aspect of this use case. For this, a reservation port type (interface) would be needed that could then be folded in to an appropriate JSDL extension to support the reservation. The back end BES implementation would of course need to be able to handle this JSDL extension and the reservation itself (though its not necessarily the case that the BES implementation has to manage the reservation, merely be aware of and work with it).

From a security perspective, existing OGSA security profiles and specifications cover most of the use case.

11. GROMACS-based Molecular Dynamics in Bio-molecular Systems

11.1 Summary

GROMACS [8] is a classical molecular dynamics application designed for simulations of large biomolecules. The code is Open Source. The application is highly popular among biophysicists, being the fastest Molecular Dynamics (MD) program. Application uses MPI or PVM to parallelize the jobs.

Job duration varies from short (few hours) to very long (few weeks). A typical parallel run lasts 2-5 days. Long simulations can be broken up into several jobs. Memory requirements range from a few up to hundreds of megabytes. The application can be pre-installed on a compute element, or can be compiled within the job scope. Compilation requires FFTW libraries.

Since MD tasks using GROMACS require massive computing resources, operate with large amounts of data, and often involve distributed communities, scientists actively seek to deploy GROMACS over the vast Grid resources. There are however several obstacles to that, as MPI support in Grid middlewares is currently rather poor.

11.2 Customers

The consumers of the GROMACS technology are:

- Scientists involved in MD simulations;
- Grid resource providers.

GROMACS usage is widespread across European HPC and cluster computing resources, and has had test implementations in Grid environments as well.

11.3 Scenarios and Applications

A user defines a job that processes a number of input files and produces a number of output files. Grid resource discovery and matchmaking services are expected to locate an optimal resource that is capable of accommodating the job and authorizes the user.

The job description and eventual input files are transferred to the Grid execution service. If input data are defined as references to a 3rd party storage, a Grid tool or service is expected to locate the data and make the available to the job (pre-stage).

On the compute element, GROMACS performs necessary preprocessing and then runs the main program that in turn produces the output files. Output files are transferred by Grid tools or services to a Grid storage facility and are made available for further analysis or additional processing by users, but not necessarily in the same workflow.

Input files are small, while output files size depends on the type and length of the job. Typically an output file size can be up to a few gigabytes, however there are no size limits.

11.4 Involved Resources and Production Grid Infrastructure

A typical example is the Finnish National Grid infrastructure M-grid [9] and HPC resources at the Finnish IT Center CSC [10]. M-grid is a Grid built of HTC resources distributed across the country, based on ARC [11], while CSC is a traditional HPC service provider. Both offer GROMACS environments for scientists, and uniform access to HTC and HPC resources is quite important. Another example is the GROMACS adaptation by the Ukrainian Academic Grid [12], which is also based on ARC.

Provided Grid resources include: computing services, storage services, authorization services and information services. In order to perform tasks, the following requirements apply to the infrastructure:

- Resources and services advertisement is necessary for discovery and matchmaking information, such as queue length, memory limits, application software availability, MPI support availability, hardware architecture details and so on;
- Input data files are available, either from the user's workstation or from a storage service, or both;
- Storage service exists for output files;
- Since the application preferably runs in parallel, MPI software on compute elements is required;
    - If application software is to be compiled, it requires the FFTW libraries and C compiler.

11.5 Security Considerations

GROMACS as such has no special security requirements, but since analysis involves file movement and re-usage by several scientists, proper authorization for data access is required. Trust and credential delegations are also required, when data movement is executed by dedicated services. In addition, parallel execution on HPC resources as within CSC demands strong authorization methods as well.

11.6 Performance considerations

The jobs are often very resource-consuming, and may last several days. This implies rather relaxed requirements on Grid services, speed-wise, as they are expected to cause only a minor overhead. The key factor in increasing processing speed is the provision of more computing power per job, which can be achieved if all resources offer same interfaces.

In order to maximize resource usage, it is very important not to occupy CPUs with non-computing tasks, such as data transfer.

11.7 Use Case Situation Analysis and PGI Expectations

Support for MPI tasks, like those executed by GROMACS, in Grid environments in general is rather poor: neither information systems, nor execution services offer adequate functionalities. Many Grid solutions also suffer from inferior data handling. PGI is expected to extend existing standard specifications such that they accommodate GROMACS use case.

Attractiveness of Grids for GROMACS users lies primarily in potential availability of several resources to choose among, all accessible in a uniform manner (e.g., SSO), and all offering identical (or near identical) execution environment, such that users don’t have to adapt their workloads for each individual resource. Therefore, PGI is expected to deliver common standards for:

- authorization and related security aspects,
- publishing information relevant for GROMACS services discovery, such as e.g.:
    - presence of MPI support as such
    - details of the software environment
    - hardware configuration
- execution service interface capable of
    - supporting data-intensive tasks
    - supporting MPI tasks
- job description language, capable of specifying GROMACS workflows, including:
    - required input and output data locations,
    - required application software,
    - required resource usage (CPU, memory, disk space, connectivity) taking into account MPI aspects

12. Multi-Grid Drug Discovery Workflow

12.1 Summary

In the last couple of years, many e-science infrastructures have begun to offer production services to e-scientists with an increasing number of applications that require access to different kinds of computational resources. Within Europe two rather different multi-national e-science infrastructures evolved over time namely Distributed European Infrastructure for Supercomputing Applications (DEISA) [13] and Enabling Grids for E-SciencE (EGEE) [14] that in turn both started the transition process towards the Partnership for Advanced Computing in Europe (PRACE) [15] and the European Grid Infrastructure (EGI) [16]. DEISA provides access to massively parallel systems such as supercomputers that are well suited for scientific applications that require many interactions between their typically high numbers of CPUs/cores. EGEE on the other hand provides access to a world-wide Grid of smaller clusters and PC pools that are well suited for farming applications that require less or even no interactions between the distributed CPUs. While DEISA uses the HPC-driven Grid technology UNICORE, EGEE is largely based on the gLite Grid middleware optimized for farming jobs and embarrassingly parallel applications.

In [17] Riedel et al. describes studies related to the improvement of a multi-Grid drug discovery workflow that is developed in collaboration with the WISDOM initiative since 2006. WISDOM aims at developing new drugs for curing Malaria. In the earlier days WISDOM scientists relied on the EGEE e-Science infrastructure for large-scale in-silico docking methods and their scientific workflows. Recently we tried to augment the drug discovery process with a complementary validation using molecular dynamic techniques on highly scalable supercomputers within the DEISA e-Science infrastructure. Hence, the first stage of the workflow should be executed on the EGEE in infrastructure, and recently EGI, resources and the second stage should use DEISA. More recently, this workflow and related studies contributed to a ‘design pattern for e-science’ [40].

To realize this workflow, we initially tried to use early adoptions of OGSA-BES in both middlewares, namely CREAM-BES and UNICORE-BES in order to take advantage of this early interoperability between both of these production Grids. Nevertheless, we experienced several shortcomings of using OGSA-BES in production setups for this particular use case. The experience will be described in this use case.

12.2 Customers

The customers of this use case raise the demand to have only one client technology that is capable of accessing infrastructures driven by HPC needs (i.e. DEISA/PRACE) as well HTC (i.e. EGEE/EGI and possibly NDGF as well). These demands are shared among many bio-informatics communities that actually seek to perform computational bioscience without the need to understand the complexity of the underlying e-science infrastructures or computational paradigms.

In general, these research groups are geographically distributed, similar to the way the resources of the infrastructures are distributed. The WISDOM initiative in particular is already an established community of scientists that continue their research and looking forward to have interoperable infrastructures not only in Europe, but also between Europe, US, and other continents like Asia and so forth.

12.3 Scenarios and Applications

The primary scenario of this use case is a multi-Grid workflow including two stages for increasing drug discovery efficiency. While the first stage requires HTC resources, the second requires HPC systems providing as much computational time and scale as possible.

In more detail, in step one, the scientific applications FlexX [18] and AutoDock [19] are used that both are typically provided on several resources within the EGEE infrastructure accessible via gLite. We foresee that this will continue to be the case transition to the EGI infrastructure is completed. Nevertheless, the output is only a list of best chemical compounds that are potential drugs and thus not the final solution.

To improve efficiency of this computational workflow, a scientific method developed by G. Rastelli et al. [20] is to use molecular dynamics (MD) computations to refine the aforementioned best compound list. While this step was initially done on the EGEE e-Infrastructure, there has been a lot of potential to use the scalable Assisted Model Building with Energy Refinement (AMBER) [21] molecular dynamics package within the DEISA e-Science infrastructure. Recently we also explored whether the MD package NAMD [22] might be more appropriate since it scales better and thus might be a better option for the second workflow step. The scaling is significantly important since it increases the accuracy of the docking results by improving the compound simulation over time what in turn is very computationally intensive.

12.4 Involved Resources and Production Grid Infrastructure

The geographically distributed computational resources involved are twofold and as follows:

- High Throughput Computing (HTC) resources: docking is an embarrassingly parallel task that does not require CPU/core interactions and thus can be executed with efficiency on HTC-driven infrastructures. Initially, the EGEE infrastructure provided resources, but we foresee that this role is taken by EGI once its transition process is completed. The Grid system here is gLite, because it provides brokering capabilities and is deployed on EGEE/EGI.
- High Performance Computing (HPC) resources: molecular dynamics are very computational-intensive and although they can be performed in a non-parallel manner statistics clearly indicate that massively parallel executions bear a lot of advantages for these kind of applications (i.e. speed-up). Initially, we used the DEISA resources, but we foresee that this role is taken by PRACE once corresponding future proposals for getting computational time is granted. The Grid system here is UNICORE, because it provides many necessary HPC features and is deployed on DEISA/PRACE.

12.5 Security Considerations

While access to bio databases and input files of the workflows (i.e. ligand data, etc.) is of minor importance, this use cases raises high demands in terms of securing access to the computational infrastructures. In this use case, scientists need to access two different complementary infrastructures using the same credential set. Apart from the usual full end-entity X.509 usage, the security should be enforced in that manner that attributes stating projects or VO memberships are equally interpreted in both infrastructures. The technology to use here can vary using either X.509 proxies with attribute extensions or SAML assertions carrying attribute statements. Initially, a SAML-based VOMS has been used.

Both of these aspects are expected to be covered PGI standardization activities in order to enable a secure multi-Grid workflow. Also delegation is required for data-staging and for the computational executions within the EGEE/EGI infrastructure (i.e. brokering). Here we require mechanisms that are compliant with SSL/TLS (i.e. https) instead of the previously used Grid Security Infrastructure (GSI) connections (i.e. httpg).

12.6 Performance considerations

The performance requirements of this use case are not strict. It is expected that the execution time is much larger than the actual submission time. Therefore, we rather see a priority in having important necessary concepts supported (cp. 1.7) rather than having a very high performance system with less functionality than required. Performance is thus a second priority in this use case while functionality in the services and schemas matters most.

12.7 Use Case Situation Analysis and PGI Expectations

This section lists what use case element concepts are relevant and thus represent a high priority of this use case. It also briefly mentions to what extent existing standards are currently satisfactory and unsatisfactory. We therefore list concepts that need to be supported in tuned standards that are expected to be delivered by PGI and explain shortly possible 'tuning' aspect of them in the following table. Many of the PGI requirements are extracted from these following missing concepts that have been partly earlier published in [40].

Concept name	Description
Production Grid-driven realistic reference model based on open standard	Although we used several standards in this drug discovery use case (OGSA-BES, JSDL, GLUE2, GridFTP, security profiles, etc.) their usage as a whole ecosystem, so to say, was rather unclear. This mainly includes computing, data, security standards as well as information flow aspects and standards. A reference model with a reference architecture and profile that considers standards would be important.
Secure Interaction	Connections between Grid technologies should be only based on normal SSL/TLS (i.e. https) connections avoiding the need for dedicated Grid technology protocols as seen previously with the Grid Security Infrastructure (GSI) (i.e. httpg). The concept of end-user rights delegation should be not strictly bounded together with the connection mechanism (as done within GSI).
Grid Application Improvements	Grid application job descriptions satisfied basic needs in this use case but were not satisfactory enough to describe an application in this multi-Grid setup. Some improvements covering but are not limited to application types classification (e.g. parallel, workflow element, etc.), application type refinements (e.g. pre-installed, submitted, etc.), revised application executable definition, application software statements, application family extension (e.g. LIBRARY), application software requirements, application output joins, etc.
Application Execution Adjacencies	In this workflow, we had several challenges in the execution environment itself. Thus we need better support for scientific application executions with standard-based information aspects on the lowest possible level (i.e. resource management system level), including common environment variables, execution modules as well as module characteristics, etc.
High Performance Computing Extensions to open standards	While runs using CREAM-BES on EGEE had been relatively ok, submission with UNICORE-BES to DEISA lacked important HPC specific information. Therefore, we seek to submit and execute applications more efficiently than currently with GLUE2, JSDL, or OGSA-BES. For instance, required aspects are support for network topologies (torus, global tree, Ethernet, etc.), shape reservations (x X y X z), network information enhancements, available shape characteristics, high message support, task/core mapping definitions, available task/core mappings exposure and re-use, etc.

Sequence Support for Computational Jobs	An analysis of lessons learned obtained from the WISDOM use case leads to specific missing features encountered during the production Grid interoperability with respect to the support of automatically started pre- and post-processing functionalities within JSDL using different application execution modes. AMBER, for instance, consists of a set of applications (~80 executables), where some of them are used to transform input data in a suitable format for production runs and/or transform outputs in several other formats necessary for further analysis. Of course, these transformation and short running pre-processing steps should be executed in a serial mode, while the actual corresponding molecular dynamic simulation is executed in a parallel mode. Pre-job sequences (pre-processing, compilation), Post-job sequences (post-processing).
Manual Data-staging support	A careful analysis of many lessons learned from this production cross-Grid application use case that takes advantage of EGEE and DEISA resources, revealed that in many cases the scientists require a more flexible mechanism during data-staging processes in order to better coordinate distributed data and computation. This is true, irrespective of whether data is transported to where the computational resource resides, or if computation is decomposed and job submissions are performed towards the location of data or even a hybrid of both methods is adopted. One example was the careful manual data input selection (aka manual data-staging) from the outcome of the EGEE workflow step in order to use only good results for the time-constrained workflow step in DEISA. This includes a feasible access to the working directory (i.e. sandbox, etc.) in order to check intermediate results.

Table 2: Required concepts to improve interoperability by PGI profiles and/or specifications.

13. RISM-FMO Coupled Simulation

13.1 Summary

Analyzing multi-scale and interdisciplinary problems is becoming increasingly important in nano-science fields, in which a simulation couples complex physical phenomena in different spatial and temporal scales. For instance, in drug design, analyzing a target molecule's electronic structure in a solvent involves extensive use of coupled simulation methodologies. The molecule's electronic structure based on a molecular orbital method is combined with solvent distribution calculation using statistical dynamics for macro regions. The coupled simulation is expected to solve large-scale problems with a high degree of physical accuracy by combining simulation components associated with different kinds of physical models and numerical methods.

The requirements for the grid infrastructure in this use case are as follows:

- The underlying grid must co-schedule multiple simulation code by discovering and co-allocating appropriate computer resources in the grid.
- Coupled simulation code must exchange data effectively between the parallel components that are programmed with very different underlying physics, divergent styles of parallel programming, and varying data representations.
- The underlying grid must absorb such differences in an automated fashion—that is, the system should support semantic transformation between synonymous physical quantities in different representations. Co-allocation of heterogeneous architecture computing resources

13.2 Customers

The research group of this use case is geographically distributed in Japan. They have the demand for using heterogeneous architecture and large scale computing resources for running their applications effectively. Each application communicates with other applications using GridMPI. Therefore, the customers have a demand for using a high-bandwidth/low-latency dedicated network between geographically distributed computing resources.

13.3 Scenarios and Applications

Figure 1 shows the scenario of RISM-FMO application. This application consists of four types having five processes (Figure 2). These processes should be run simultaneously. The system has to provision computing resources on the same time slot. To support this, the execution service of RENKEI/NAREGI system has two functionalities in it service interface, advanced resource reservation, and reserved schedule advertisement. Information in context can be found at [23].

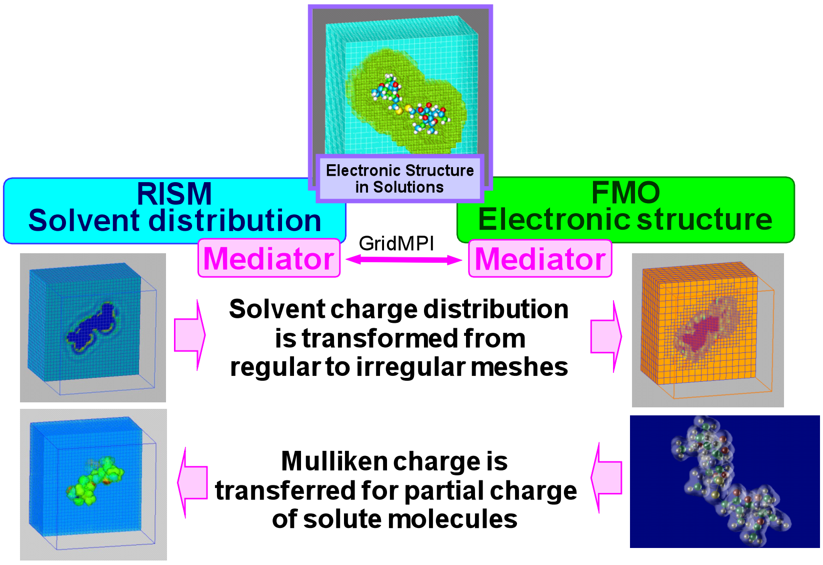
Electronic Structure in Solutions
RISM
Solvent distribution
FMO
Electronic structure
Mediator
GridMPI
Mediator
Solvent charge distribution is transformed from regular to irregular meshes
Mulliken charge is transferred for partial charge of solute molecules

Figure 1 Scenario of RISM-FMO application

The working scenario is as follows:

A) Preconditions

- RENKEI-IS (information service) always collecting resource information from RENKEI-GridVM(Execution service) using secure connection
- Server certificates for all services are issued by a trusted CA and always checked CRL
- Each user have X509 user certificate issued by a trusted CA and also have proxy certificate for job submission
- RENKEI-SS (meta-scheduler) has advanced reserved resource schedules of all computing resources that SS is managing

B) Job execution

- User submit the RISM-FMO job using RENKEI-Portal and WFT (WorkFlow Tool)
- The SS parses the JSDL and look for some GridVMs that have the following conditions: (a) Resource properties are matched with the job requirements that described in the JSDL; (b) The resources are not reserved for the execution time slot
- The SS try to make reservations for the resources
- If the SS could get the time slot of the resource, it delegates a short life time proxy certificate to GridVMs, that manages the reserved resources, then make commit and submit the jobs to it with secure interface
- The GridVM checks proxy certificate with CRL etc.
- If it passed, GridVM parse the job (JSDL) and make a script for LRMS, then submit job to the LRMS
- At 2/3 of the life time of the proxy certificate, it is checked again.
- If it passed, the proxy certificate is renewed by the RENKEI-Renewal Service

C) Post conditions

- When the execution finished or failed, GridVM generates a notification to SS
- When the SS receives the notification, SS forward it to the Portal and WFT

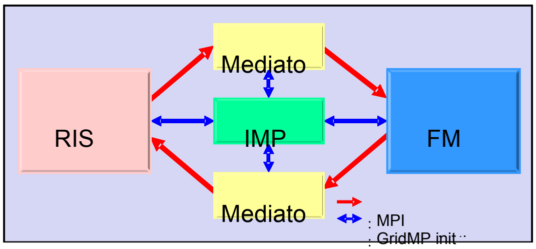

Figure 2 Configuration of the RISM-FMO icon

13.4 Involved Resources and Production Grid Infrastructure

The computing resources are heterogeneous and geographically distributed. The RISM application requires a large scale clustered SMP system. Any CPU architectures are acceptable, but each node satisfies the following conditions: Unix/Linux OS, more than 16 cores, and more than 32GB memories. The FMO application requires a large scale cluster system which satisfies Unix/Linux OS, x64/x86 architecture, more than two cores, more than 4GB memories and more than 400 nodes. The requirements for other applications are: Linux, x86/x32 architecture, few cores, and 4GB memories.

A customer installed NAREGI infrastructure provided these resources.

Production Grid services in this context are services in the context of job execution service (such as GridVM in NAREGI), computing resource broker and work load manager (such as SuperScheduler in NAREGI).

13.5 Security Considerations

The following security aspects are required:

- User authentication and authorization mechanisms based on X509 certificates and VOMS attributes
- Delegation mechanism for allowing services for job execution and data transfer management operations on behalf of the user
- Proxy certificate renewal mechanism for using short life time proxy certificates
- Accounting system for charging the resource usage

13.6 Performance considerations

The applications in this use case are classified as HPC applications. These applications will run long time and consists of only five activities, therefore the performance of grid services are not required as high.

13.7 Use Case Situation Analysis and PGI Expectations

This use case expects the following functionalities in terms of PGI specification/profile. The following list involves expectations that require modification for some basic specifications such as OGSA-BES and JSDL.

- Advanced resource reservation
- Inform advanced reservation (scheduling) table for meta-scheduler and work load manager
- GridMPI support
- Inform allocated worker nodes list (for each GridMPI activity)

14. Data intensive computation job processing on cluster-based Grids

14.1 Summary

This use case is among most common single scenarios executed on European scientific Grid infrastructures that grew from needs of the High Energy Physics (HEP). In general, it reflects a subset of most challenging HEP tasks, but is also applicable to other sciences dealing with huge amounts of data that need to undergo complex transformations (sometimes called data reduction).

Combination of several aspects makes this use case a distinct scenario:

- Large amounts (up to dozens of Gigabytes per single workflow) of input and output data units containing independent information (allowing for embarrassing parallelism);
- Input and output data are stored over a wide network area;
- Long processing times per individual process (up to several days);
- Complex application-specific transformations with many failure modes (application codes measured in Gigabytes);
- Usage of heterogeneous Linux clusters as the computational resource (mixture of incompatible operating systems and middleware systems);
- Highly complex mixture of administrative domains and policies for execution and storage services, as well as for users' affiliations (thousands of Virtual Organisation members using hundreds of potential service providers in dozens of countries).

This use case is positioned as the "ARC Use Case" because the ARC middleware [24] was originally designed to address this scenario as much as possible, particularly in order to satisfy the needs of scientists working for the ATLAS experiment at CERN [25]. More details on the ATLAS usage scenario and performance of ARC in this context can be found elsewhere [26].

14.2 Customers

Several kinds of customers can be identified in this use case:

- End-users, i.e., scientists seeking to obtain results of data processing, often authoring data transformation software, typically affiliated with a Virtual Organisation;
- Scientific application developers, i.e., the experts providing application-specific software layers that expose to the users only those aspects of the computing infrastructure which have relevance for the immediate scientific task;
- Resource providers, i.e., administrators and managers of execution, storage and information services, as well as other services eventually involved.

In practice, the first two groups are often indistinguishable, which results in rather monolithic application-specific software tools that penetrate across infrastructure layers.

14.3 Scenarios and Applications

The generic scenario can be outlined as such:

A Virtual Organisation member produces a task/workflow characterization by specifying a large set of input data, transformation to be performed, its performance requirements, and desired output data destination. The transformation may be characterized as a complex set of processes involving different software – a workflow. Due to the large amounts of independent data to be processed, the task may be described as a large set of independent sub-tasks. A software tool (either an application-specific interface, or a generic Grid tool) queries available information about the Grid infrastructure, matches the task requirements to an optimal execution service, and submits the task to the selected execution service (or a set of such, in case of multiple sub-tasks). The execution service interprets the submitted job description and proceeds with locating and fetching the requested input data, typically from a remote storage, and on behalf of the user in order to comply with access policies. Further, when all input data are made available locally, the execution service launches the transformation itself. Since the transformation may take very long time, users and/or tools query the execution service in order to assess the task progress (or lack of such, in case of a subtle internal transformation failure). Upon task completion, the newly created data is uploaded by the execution service to a (typically, remote) storage resource requested in the task description, provided the user is authorized (e.g. as a member of a Virtual Organization).

This high-level scenario has several implications which need to be described in more details:

- Complexity of transformations leads to multiple failure modes; some of them are not critical and can be avoided when spotted at sufficiently early stages of execution (e.g., typos in output file names). Since such failure modes are very application-specific, they cannot be identified by generic Grid tools; therefore, users resort to regularly reading standard output or error logs of the jobs at the actual execution location, and even internal execution service logs pertaining to specific jobs.
- When a devious task flow is thus identified, it may be too expensive to cancel and re-start the job; some application failure modes can be quickly rectified by modifying specific files at the execution location, without interrupting execution. A very common example is upload of a new proxy certificate when the original one is about to expire.
- Even though some failures may be avoided when the problem is spotted early enough during execution time, many failures still lead to execution termination. Still, many of them are caused by temporary recoverable problems: most frequently, temporary storage service unavailability. To recover from such problems, it is sufficient to re-start the job from the point where it failed. This leads to a more efficient resource usage as opposed to the conventional full job restart, given large amounts of data involved and complexity of transformations.
- Lengthy execution times frequently mean that a user physically cannot monitor job status during some periods of time (e.g., holidays, weekends). Nevertheless, application-specific tools and regulations often require users to perform specific actions on finished (failed or succeeded) jobs, such as e.g. error log retrieval or such. In order to accomplish this while allowing users to take a break from the job, users have to rely on persistency of information about Grid jobs available through execution services.
- Given large amount of input data, on one hand, and large amount of potential execution targets on another, the process of job submission can take rather long, up to several minutes per single job. Meanwhile, a typical task/workflow consists of thousands of such jobs. Normal users can not afford waiting days until their jobs get submitted. In this situation, preference is given to Grid services that minimize submission time.
- Transformations typically consist of a large number of separate procedures, and are often steered by the means of auxiliary files which are created by the users in the process of task definition. The case is much more complex than a parameter sweep, as a steering file may include hundreds of application-specific parameters and even instructions. Naturally, no generic Grid job description language can provide for such specific instructions, thus such files are provided as input to the job. However, contrary to the input data sets, these files are not permanently stored at storage facilities, but are dynamically created by users at their work stations, and therefore are transferred from the work station to the execution service alongside the job description.
- The job life cycle proceeds through several stages, each of which can take hours or even days. Large number of available resources of different kind gives certain flexibility, allowing users and their application-specific to steer the overall scientific analysis process. For example, if many jobs are stuck in a specific stage before execution, it may be beneficial to cancel them and re-submit to a better performing resource. For efficient steering of this kind, users rely on detailed human-readable information about underlying activities provided by the execution service.

14.4 Involved Resources and Production Grid Infrastructure

The use case involves all basic resources forming modern Grid infrastructures serving HEP and similar tasks, namely:

- Execution services;
- Client tools;
- Data storage and management services;
- Information services;
- Authorisation services;
- In some cases, intermediate services involved in job handling between client tools and execution services (e.g., meta-schedulers).

An example of a production Grid infrastructure for which this use case is very common is the Worldwide LHC Computing Grid (WLCG) [27]. WLCG brings together resources powered by gLite [28], ARC and solutions provided by the Open Science Grid (OSG) [29]. The Nordic DataGrid Facility (NDGF) [30] is another production Grid infrastructure where majority of applications are described by this use case, and it exclusively relies on ARC to provide execution services.

14.5 Security Considerations

Security considerations are quite strict, requiring full control over users' access rights to computing resources, information and data. Data encryption however is not used. Involvement of a variety of intermediate services (e.g. for data transfer) imply the necessity of delegation mechanisms.

It must be pointed out that in most cases access control is enforced on the Virtual Organisation (VO) level; that is, access to resources and data is normally granted not per individual user, but per VO. In many cases, data and even jobs are collectively "owned" by a VO, such that any VO user (or everybody having the specific "role") can manipulate jobs or data.

Still, for the reasons of traceability and accountability across administrative domains, typical infrastructure policies require that identity of each individual user has to be known to all involved resources participating in activities related to job defined by this user. Access control to job-related data and information must be handled reliably. Sharing of information among various identities (e.g., VO members in the same role) must comply with the rules set by the user.

14.6 Performance considerations

Some aspects of the described scenario may lead to severe performance penalties when addressed in a conventional manner. For example, intensive resource polling by user tools lead to denial of service. Clearly, implementations must aim at smallest performance impact.

From the end-user perspective, responsiveness is the key performance parameter – be it job submission time or job status query execution time.

From the resource providers' perspective, decreasing and optimizing amount of data being transferred between resources is important for providing better throughput and quicker turnaround. Obviously, quite often both requirements cannot be achieved simultaneously, thus ways of balancing between them need to be considered.

14.7 Use Case Situation Analysis and PGI Expectations

The use case is currently largely addressed by the ARC middleware, and to a large extent – by the gLite middleware as well. However, submission of this kind of tasks across different middlewares is very problematic due to absence of common interfaces. On application level in the infrastructures like WLCG this lack of interoperability and desired functionality is solved in the very unorthodox manner: all HEP experiments deploy agent-based (a.k.a. pilot) jobs, which represent dummy payloads without data and with a trivial "transformation" which has a task of fetching the actual payload from an application-specific service (job pool). While being a very convenient and flexible framework for applications, this approach creates multiple problems for resource owners, and does not help application developers at all; neither does it promote standardization.

By delivering a common execution service interface and job description complete with well-defined and flexible data staging capabilities, increased user friendliness and better fault tolerance, PGI will reduce the burden on application developers, which in turn will attract more user communities to the Grid.

15. European Grid Infrastructure

15.1 Summary

The European Grid Initiative [16] (EGI) is a major European Union effort to create an e-Science infrastructure making use of a large variety of distributed computing technologies and to strengthen Europe's position in the area of distributed computing infrastructures. In order to support this initiative, the EGI-InSPIRE project was created, bringing together all national Grid infrastructures in Europe. In order to support this infrastructure from the technological point of view, EGI will rely on a stack of certified middleware components delivered by different middleware providers – the Unified Middleware Distribution (UMD). It is of ultimate importance for these components to comply with same standards in a uniform manner.

The grid software used in Europe, : gLite [28], ARC [11], UNICORE [31], Globus [32] and dCache [33]) share some approaches, but there are still substantial differences, as historically the middlewares were developed before standards were defined, and often rely on proprietary solutions. In order to harmonise the glite, ARC, UNICORE and dCache software, another project has been established, the European Middleware Initiative [34] (EMI). EMI is expected to be one of the major technology providers for EGI.

A substantial number of research infrastructures in Europe rely on technologies found in Globus Toolkit. In order to support European Globus users, and to ensure inclusion of Globus-based resources into the common European Grid Infrastructure, yet another project is created, the Initiative for Globus in Europe [35] (IGE).

In addition to the HPC- and cluster-based infrastructure components, the European Grid infrastructure is expected to have a desktop-based segment, still seamlessly included into the operational structure. This segment is supported through the EDGI project [36], which offers technology solutions specific for desktop computing, bridging gaps between middleware components from other providers.

The EGI use case thus requires standards-based interoperability of the mentioned technologies (gLite, ARC, UNICORE, dCache, Globus and EDGI), and implies existence of mandatory standards inventory in order to guarantee inclusion of eventual new technologies into the infrastructure, provided they comply with defined standards. This in turn makes it also possible to prevent vendor-locks so that end-users of EGI can choose their technology of choice where possible.

15.2 Customers

Stakeholders in the use case:

- Potentially, all European academic[6] researchers engaged in e-Science, including:
  - End-users with little or no software skills,
  - Application developers
- Potentially, all European providers of research computing services, including:
  - Managers, policy-makers and decision-makers, not necessarily familiar with technology
  - Technical personnel, with varied expertise in Grid technologies
- European Grid technology providers, primarily through EMI, IGE and EDGI projects,

It is expected that many of these stakeholders will be represented by EGI.eu, including the National Grid Initiatives (NGI) within Europe. The resulting infrastructure is by definition multi-national and spreads across very different administrative and legislative domains. As a consequence the procedures and policies established by EGI must be independent of technologies used by resource providers, yet retain an integrated and interoperable infrastructure for multi-disciplinary use.

End-user applications are expected to cover all public research areas, with possible reach to commercial applications.

[6] Commercial research for non-commercial use is acceptable

15.3 Scenarios and Applications

The very high-level scenario is provision of arbitrary computational and/or storage resources to an arbitrary group of researchers (or an arbitrary individual researcher). These services offered by the infrastructure provider may be a mix of generic services applicable for all users, or specific services for specific communities. The generic services within EGI will be supported by resource providers or hosted centrally as applicable.

Within EGI the common services supported by the infrastructure will be defined by the Technology Coordination Board (TCB) in consultation with the users and resource providers. These common services will be verified by the EGI.eu Technology Unit and their effective operation in production will be demonstrated through a staged rollout into wide-scale production use. Procedures describing such as resource provision, access, monitoring, accounting, auditing, etc must be independent of the used technology, such that a researcher can execute the same application over any European resource as long as it provides the required functionality, while resource providers can rely on common operational tools to support the infrastructure.

A key scenario is the ability to submit a unit of work (an application with associated input files being retrieved from a specified resource) to a computing resource for execution with any resulting output data files being transferred to a specified resource for later use.

Here “technologies” refer to those found in EMI, IGE and EDGI projects and eventual other standards-compliant solutions in future.

15.4 Involved Resources and Production Grid Infrastructure

EGI directly involves all national Grid infrastructures in Europe and their resources, including academic networks. Active interoperations with non-European computing infrastructures, such as e.g. Open Science Grid [37] are also foreseen.

15.5 Security Considerations

The overall size of the combined EGI resources makes the infrastructure a very attractive target for malicious exploits. Size and international scale impose non-conventional strict security requirements, the relaxation of which may lead to a collapse of the infrastructure. Security and policy issues in EGI will be largely inherited from pre-existing international infrastructures; currently, security and policy documents produced by the Joint Security and Policy Group [38] are the basis for the operations. There, fundamental requirements are traceability of all activities while respecting privacy of users and protecting sensitive information. Extreme complexity of the infrastructure and frequent involvement of several distributed services in a single workflow in practice requires credential and trust delegation technologies.

15.6 Performance considerations

Currently, key requirements are high reliability and efficiency of the infrastructure, even if they come on expense of performance. Performance can be measured in different terms, depending on applications: speed of task execution, efficiency of resource usage, integrity of transferred data, accessibility of monitoring information, etc. It is desirable to have a balanced system, where performance penalties in one criterion are balanced by gains in another.

15.7 Use Case Situation Analysis and PGI Expectations

As EGI encompasses all possible services used by national Grid infrastructures across Europe, it requires standardization on all levels: computing interfaces, data management, information, security etc. At the moment, arguably the only commonly implemented standard is SRM 2.2 [39]. PGI is expected to deliver common profiles for all other relevant services.

16. Conclusions

The use cases have been provided by members of different production Grids and technology providers covering a wide variety of common application demands. While there is partly a high diversity there is a lot of common ground for similar requirements as well. For instance, many of the use cases raise a demand of an improved way of managing and executing High Performance Computing (HPC)-oriented and parallel applications with open standards. Other use cases clearly raise the demand to improve the data management and data-staging functionalities as well as having a strong security setup in place, including delegation of rights. Others mention the necessity of improved application descriptions and support in open standards. Here we can expect a lot of overlaps in terms of requirements arising from different use cases.

Until today, the use cases surveyed within this document have led to a wide variety of requirements for a set of profiles and specifications that are able to address production Grid demands. Clearly we can see the demand for prioritization among all the resulting requirements from these use cases focusing on the critical and vital important requirements of each stakeholder as much as possible.

Finally, the complexity of the agreement process in terms of the set of resulting requirements from these use cases brings also a fundamental chance for a high adoption rate of PGI specifications. We can foresee that the regular participation of major production Grids and highly relevant technology providers will lead in turn to a significant adoption by them thus laying the foundation for a broad usage in production. Most notably, the mix of production Gird infrastructures and their major technology providers make it possible to provide a realistic set of PGI profiles and standards that will be deployed and used in production for a long time.

17. Editor Information

Morris Riedel
Jülich Supercomputing Centre (JSC), Forschungszentrum Juelich (FZJ) GmbH, Juelich, Germany
Email: m.riedel@fz-juelich.de

Johannes Watzl
Ludwig Maximillians University Munich, Germany
Email: watzl@nm.ifi.lmu.de

18. Authors

Emmanouil Paisios
Leibniz Rechenzentrum (LRZ), Munich
Email: Emmanouil.Paisios@lrz.de

Luigi Zangrando
INFN – Padova
Email: luigi.zangrando@pd.infn.it

Etienne Urbah
LAL, Univ Paris-Sud, IN2P3/CNRS
Email: urbah@lal.in2p3.fr

Andrew Grimshaw
University of Virginia
Email: grimshaw@virginia.edu

Mark Morgan
University of Virginia
Email: mmm2a@virginia.edu

Steve Crouch
University of Southampton
Email: s.crouch@omii.ac.uk

Morris Riedel
Jülich Supercomputing Centre (JSC)
EMail: m.riedel@fz-juelich.de

Oxana Smirnova
University of Lund
Email: oxana.smirnova@hep.lu.se

Ivan Degtyarenko
CSC
Email: Ivan.Degtyarenko@csc.fi

Kazushige Saga
National Institute of Informatics
Email: saga@grid.nii.ac.jp

Aleksandr Konstantinov
University of Oslo
Email: aleksandr.konstantinov@fys.uio.no

Steven Newhouse
European Grid Initiative (EGI)
Email: steven.newhouse@egi.eu

19. Contributors

We gratefully acknowledge the contributions made to this document to the various authors listed in the previous section. But apart from the authors, we would like to thank the following contributors:

Use-case information related to the Virtual Physiological Human (VPH) was provided by Peter Coveney, Abhinav Thota, Shantenu Jha and Andre Merzky.

20. Acknowledgments

We are grateful to numerous colleagues for discussions on the topics covered in this document, and to the people who provided comments on the public drafts.

21. Intellectual Property Statement

The OGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the OGF Secretariat.

The OGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the OGF Executive Director.

Disclaimer

This document and the information contained herein is provided on an “As Is” basis and the OGF disclaims all warranties, express or implied, including but not limited to any warranty that the use of the information herein will not infringe any rights or any implied warranties of merchantability or fitness for a particular purpose.

22. Full Copyright Notice

23. References

[1] Virtual Physiological Human network of excellence, http://www.vph-noe.eu/

[2] SAGA, http://saga.cct.lsu.edu/

[3] EGI SPG (Security Policy Group), https://wiki.egi.eu/wiki/SPG

[4] GLUE Specification v. 2.0, http://www.ogf.org/documents/GFD.147.pdf

[5] Southern California Earthquake Center Earthworks Project, http://kb.iu.edu/data/avbq.html

[6] Crystal Structure Prediction of the Organic Solid State (CPOSS) Project, http://www.cposs.org.uk/

[7] http://www.tacc.utexas.edu/RangerImpact/pdf/Science_Center_of_Storm.pdf

[8] http://www.gromacs.org

[9] http://www.csc.fi/english/research/Computing_services/grid_environments/mgrid

[10] http://www.csc.fi

[11] http://www.nordugrid.org/arc

[12] http://uag.bitp.kiev.ua

[13] Wolfgang Gentzsch, Alison Kennedy, Hermann Lederer, Gavin Pringle, Johannes Reetz, Morris Riedel, Bernd Schuller, Achim Streit, and Jules Wolfrat, ***"DEISA: e-science in a collaborative, secure, interoperable and user-friendly environment"***, Proceedings of e-Challenges 2010 Conference, Warsaw, Poland

[14] EGEE Website: http://public.eu-egee.org/

[15] PRACE Website: http://www.prace-project.eu/

[16] EGI Website: http://www.egi.eu

[17] M. Riedel, A.S. Memon, M.S. Memon, D. Mallmann, A. Streit, F.Wolf, Th. Lippert, V. Venturi, P. Andreetto, M. Marzolla, A. Ferraro, A. Ghiselli, F. Hedman, Zeeshan A. Shah, J. Salzemann, A. Da Costa, V. Breton, V. Kasam, M. Hofmann-Apitius, D. Snelling, S. van de Berghe, V. Li, S. Brewer, A. Dunlop, N. De Silva - ***Improving e-Science with Interoperability of the e-Infrastructures EGEE and DEISA,*** Proceedings of the 31st International Convention MIPRO, Conference on Grid and Visualization Systems (GVS), May 2008, Opatija, Croatia, Croatian Society for Information and Communication Technology, Electronics and Microelectronics, ISBN 978-953-233-036-6, pages 225 – 231

[18] FlexX Website: http://www.biosolveit.de/FlexX/

[19] AutoDock Website: http://autodock.scripps.edu/

[20] G. Rastelli et al., ***"Validation of an automated procedure for the prediction of relative free energies of binding on a set of aldose reductase inhibitors"***, In Bioorg Med Chem, 2007 Dec. 15, 15(24), 7865-77, Epub 2007.

[21] AMBER Website: http://amber.scripps.edu/

[22] NAMD Website: http://www.ks.uiuc.edu/Research/namd/

[23] S Matsuoka, K Saga, and M Aoyagi, "Coupled-Simulation e-Science Support in the NAREGI Grid," in Computer 41 (11), 2008.

[24] ARC, http://www.nordugrid.org/arc

[25] http://cern.ch/atlas

[26] "Performance of the NorduGrid ARC and the Dulcinea Executor in ATLAS Data Challenge 2". R.Sturrock et al., in Proceedings of CHEP 2004, eds. A.Aimar, J.Harvey and N.Knoors, CERN-2005-002, Vol. 2, 2005, p. 1095.

[27] http://cern.ch/lcg

[28] http://cern.ch/glite

[29] http://www.opensciencegrid.org

[30] http://www.ndgf.org

[31] http://www.unicore.eu

[32] http://www.globus.org

[33] http://www.dcache.org

[34] http://www.eu-emi.eu

[35] http://www.ige-project.eu/

[36] http://edgi-project.eu/

[37] http://www.opensciencegrid.org/

[38] http://www.jspg.org

[39] The Storage Resource Manager Interface Specification Version 2.2, http://www.ogf.org/documents/GFD.129.pdf

[40] Morris Riedel, ‚**e-Science Infrastructure Interoperability Guide: The Seven Steps Toward Interoperability for e-Science**', in Book 'Guide to e-Science', Springer Book Series Computer Communications and Networks, 2011