GFD-I.029
Category: Informational
Open Grid Services Architecture Use Cases

Editors:
I. Foster, Argonne & U. Chicago
D. Gannon, Indiana U.
H. Kishimoto, Fujitsu Labs
Jeffrin J. Von Reich, Hewlett Packard

March 4, 2004

Open Grid Services Architecture Use Cases

Status of this Memo

This document provides information to the community regarding the Grid use case scenarios used in the definition of Open Grid Services Architecture (OGSA) Platform components. Distribution of this document is unlimited.

Abstract

Successful realization of the Open Grid Services Architecture (OGSA) vision of a broadly applicable and adopted framework for distributed system integration requires definition of a wide variety of Grid use case scenarios of both e-science and e-business applications. Use cases described in this document cover commercial infrastructure and application topics (Commercial Data Center, Online Media and Entertainment, Inter grid), scientific infrastructure and application topics (National Fusion Collaboratory, Severe Storm Modeling, and Virtual Organization Grid Portal), essential grid technologies (Grid Resource Reseller, Service-Based Distributed Query Processing, and Workflow, Grid lite, Interactive grids) and working group use cases (mutual authorization, persistent archives, resource usage service). The list of Grid use cases presented here is necessarily incomplete. Also use cases are not described at the detail required for formal requirements.

GLOBAL GRID FORUM
office@gridforum.org
www.ggf.org

Full Copyright Notice

Intellectual Property Statement

The GGF 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 GGF Secretariat.

The GGF 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 GGF Executive Director (see contact information at GGF website).

Contents

1 Introduction

One component of the OGSA-WG's charter is

> "To produce and document the use cases that drive the definition and prioritization of OGSA Platform components, as well as document the rationale for our choices."

This document is a collection of the use case scenarios contributed by OGSA-WG participants or solicited from others. It is a companion to "The Open Grid Services Architecture Platform". This document contains fully reviewed use cases in the specified OGSA template. There is a companion use case document along with this, namely 'Open Grid Services Architecture: Second Tier Use Cases' (document id is GWD-I: draft-ggf-ogsa-tier2-usecase-2). That document contains all the incomplete, and non reviewed use cases to date.

Based on the use case documents the OGSA-WG will (a) specify, in broad but somewhat detailed terms, the scope of important services required, (b) identify a core set of such services that are viewed as essential for many Grid systems and applications, and (c) specify at a high-level the functionalities required for these core services and the interrelationships among those core services.

While these use cases have certainly not been defined with a view to expressing formal requirements (and do not contain the level of detail that would be required for formal requirements), they have provided useful input to the definition process. We expect to expand the number of use cases in future revisions of this document.

Table 1: Use cases and contributors in this document

Chapter	Title	Contributors
2	Commercial Data Center	Hiro Kishimoto, Andreas Savva, David Snelling
3	Severe Storm Modeling	Dennis Gannon
4	Online Media and Entertainment	Tan Lu, Boas Betzler
5	National Fusion Collaboratory	Kate Keahey
6	Service-Based Distributed Query Processing	Nedim Alpdemir, Norman Paton
7	Grid Workflow	Takuya Araki
8	Grid Resource Reseller	Jon MacLaren, William Lee
9	Inter Grid	Jeffrin J. Von Reich
10	Interactive Grids	Jeffrin J. Von Reich
11	Grid Lite	Jeffrin J. Von Reich
12	Virtual Organization Grid Portal	Charles Severance
13	Persistent Archive	PA Working Group of GGF.
14	Mutual Authorization	Takuya Mori
15	Resource Usage Service (RUS)	Bill Horn

2 Commercial Data Center

2.1 Summary

Many enterprises have been consolidating IT resources such as servers and storage into data centers in order to reduce the total cost of ownership. In addition, many enterprises are outsourcing or planning to outsource their IT resources and/or their management, which allows them to focus on their core businesses. Consequently, data centers need to manage several thousands of IT resources, which include servers, storage, and networks. Decreasing the management complexity and increasing utilization of these resources require an innovative GRID based resource management software, which we call a "Commercial GRID System" (CGS). All references to Grid technologies or simply to "Grids" in this use case refer to the CGS. Finally, we call a data center that implements the CGS a "Commercial Data Center (CDC)."

During the time that mainframes dominated IT, an IT system integrator could develop a controllable IT system on top of this single, solid, and homogeneous platform. The current IT system integrators, however, must use tens of different APIs on different OSes and middleware platforms, which have no consistent way to detect and respond to faults (to improve availability) or identify underlying performance bottlenecks (to meet performance targets), and thus have no consistent way to guarantee QoS. Grid-based meta-OS functionalities provided by the CGS can ease the burden of IT system integrators by enabling end-to-end QoS.

2.2 Customers

The **"Grid administrator"** is an important actor of the CDC. Strictly speaking, the Grid administrator is not a customer but a provider. However, the Grid administrator benefits from the increased manageability of the IT infrastructure provided by the Grid in the CDC. This is one of the key motivations of the CGS. Since the management of the hardware and software on the CDC is difficult and costly, the administrator demands the automation of key functionalities such as provisioning, monitoring, tuning, maintenance, error diagnosis and fault recovery on the IT infrastructure.

One requirement placed on the Grid administrator is to increase the utilization of the IT infrastructure. According to several analysts' reports, actual utilization ratio is often less than 20% for scattered resources, increasing to 70% or more when they are consolidated. Also some resources are reserved for failover and provisioning; in other words, they are not put to productive use. It should be possible to share such resources among multiple systems, with physical location not being the single determining factor whether sharing is possible or not.

The Grid increases IT infrastructure manageability thereby minimizing the number of administrators, e.g. from a few dozens to less than ten.

The **"IT System Integrator"** is a customer of the Commercial Data Center. The IT System Integrator has the difficult task of constructing heterogeneous systems. Problems include making end-to-end performance predictions and guarantees, ensuring the required level of availability is achieved (e.g., 99.99%), provisioning of additional resources to respond to unpredictable service demands (e.g., the internet spike problem), while at all time responding to frequent changes (discounts and resulting access load changes, number of products, new services, etc.).

The IT System Integrator expects to reduce the complexity of building distributed and heterogeneous systems by means of an OGSA based Grid, which provides standard and QoS-enabled meta-OS functionalities.

The IT system integrator can also use the Grid to easily create test systems (through the creation of VOs).

The **"IT business activity manager"** is another customer of the Commercial Data Center. The IT business activity manager, for example, runs a ticketing service which sells tickets to **"End Users."** The end users are actors of the CDC but not its direct customers– they are customers of the ticketing service.

At the moment only a few IT business activity managers use the CDCs. We expect that in the future hundreds of these managers would be using each data center.

The following figure depicts some of the actors described above. The data centers correspond to Real Organizations (ROs) and the IT business activities correspond to Virtual Organizations (VOs). The IT business activity managers create VOs and run their services in them, expecting that the VOs are reliable, scalable, secure, and deliver the required QoS. On the other hand, the Grid administrators manage ROs and the Grid alleviates their work.

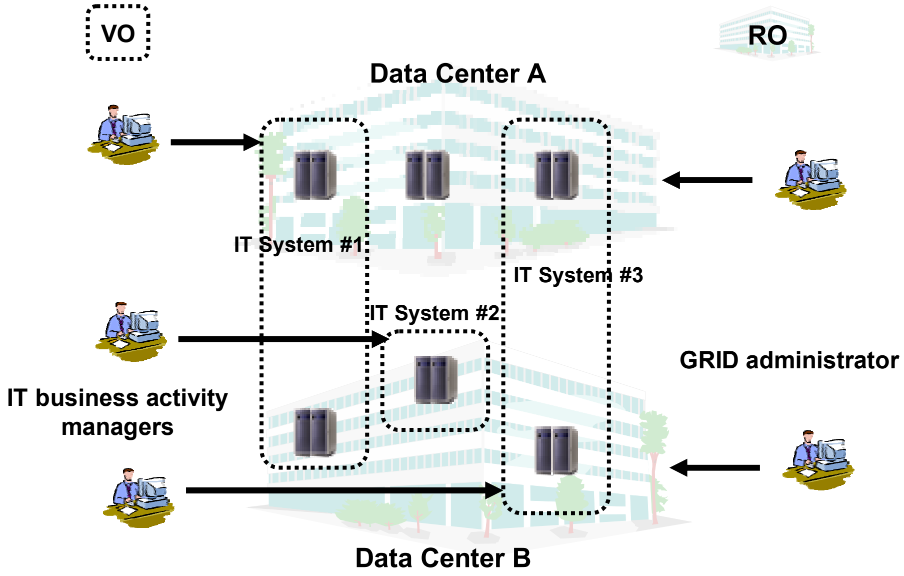


Figure 1: ROs, VOs, and customers of the Commercial Data Center

2.3 Scenarios

There are four scenarios for the Commercial Data Center.

2.3.1 Multiple in-house systems

Current in-house systems, e.g. for personnel management system, finance and accounting, order-receiving and customer relationship management (CRM), are mostly isolated. Each in-house system runs on its own IT resources and also keeps extra IT resources for high availability or in preparation for increased workload. Since the workloads are all different and peaks do not necessarily occur at the same time, there are a lot of idle IT resources.

If the Grid could manage a large part of the IT resources in the enterprise and could provide necessary resources to each in-house system on demand, extra resources needed by each system could be shared among several systems, leading to better IT resource utilization. Also, more in-house systems could run on less IT resources.

For each in-house system, the Grid makes reservations in advance, allocates hardware, deploys necessary software and data, and starts the needed applications. All these procedures are automated.

The Grid also provides autonomous management including failover and provisioning. The Grid handles many failures autonomously.

Additionally, multiple remote data centers could work together to improve scalability and availability. Undisrupted operation must be ensured even in the event of disasters such as earthquakes, fires, or acts of terrorism. Independent, but networked, data centers can be used to provide the necessary physical infrastructure.

2.3.2 Limited time commercial campaign

Corporate marketing often plans limited time campaigns, e.g. concert ticket sales, international conference registration, or sales promotion campaigns. Current systems for these campaigns require fixed IT resources, which are over-provisioned to cope with peaks in demand. Thus they need high initial purchase and maintenance costs. The Grid could provide necessary IT resources on demand and charge based on usage.

IT business activity managers can also chose the most inexpensive data centers or use multiple data centers for scalability and availability.

2.3.3 Disaster recovery

IT systems providing essential public infrastructure services, such as banking systems and air traffic control systems, require disaster recovery capabilities. Popularization of the Internet also makes many applications - e.g. popular web pages like Google, indispensable. Disaster recovery, however, has a very high cost and requires a very high level of technical expertise to build and operate.

The Grid could provide a standard disaster recovery framework across remote CDCs to these IT business activities at lower cost.

2.3.4 Global load balancing

Geographically separated CDCs can share high workload and provide scalability for applications.

2.4 Involved resources

A CDC is equipped with all sorts of IT resources including servers, storage, data, and networks. The Grid should manage at least several thousands of resources.

2.5 Functional requirements for OGSA platform

For the scenarios described above the following functions are required:

1. **Discovery**

   At first, an actor of the CDC should pick out a reference to the CDC, which he/she will use. One or more well-known discovery services are used as the first step.

2. **Authentication, Authorization, and Accounting (AAA)** [1]

[1] This function should be added to OGSA platform functionality.

When the customer submits a job request, the CDC authenticates the customer and authorizes the submitted request. The CDC also identifies his/her policies (including but not limited to SLA, security, scheduling, and brokering policies). The Grid checks if the customer has the right to perform the requests sent.

3. **Advance Reservation** [2]

   Based on the customer's request the Grid registers when to start the request processing. [3] The Grid interprets the job specification description language in which the request is written.

4. **Brokering**

   The Grid finds the most suitable resources for the requested time period (assuming a request for advance reservation). Access-control to the resources and quotas are also applied. The reservation is made and its reference is returned to the customer.

5. **Data Sharing**

   The job request also specifies required user data (databases and/or files). Data accessibility should be considered during match-making.

6. **Provisioning**

   Some time before the reservation time, the Grid begins application and user data deployment. In the case of a Java program, the Grid discovers the designated java program (jar file) and deploys it into the reserved resource. The deployment feature for Java is already well-defined and supported on most hosting environments.

7. **Scheduling** [4]

   When the reservation time comes, the Grid starts the task.

8. **Metering and Accounting**

   During job execution, the metering service keeps track of resource usage. The information is passed to the accounting service.

9. **Fault Handling** [5]

   For this use case it is assumed that the customer only needs failure notification in case his/her job encounters an error and cannot complete successfully (the fault handling procedure is designated through fault management policies).

10. **Policy**

    Several attributes should be handled as policy. A brokering policy defines resource usage quotas per customer. An error and event policy guides autonomous management including provisioning and failover.

11. **Security**

    Isolation of customers in the same data center is a crucial requirement. The Grid should provide not only access control but also performance isolation.

[2] This function should be added to OGSA platform functionality.

[3] "Request processing" and "job processing" are different. In case of advanced reservation, the request processing books resources for future use and job processing is actual job execution at the reserved time.

[4] This function should be added to OGSA platform functionality.

[5] This function is called "Fault Tolerant" in [References: 1]. In order to cover more generic functionality, the function is renamed to "Fault Handling" in this document.

For the scenario "Limited time commercial campaign," the following functions are required in addition to the above:

12. Virtual Organization

Upon the customer job request the Grid creates a VO in a data center which provides IT resources to the job. Depending on the customer's request, the Grid will negotiate with another Grid on remote CDC and create a VO across the CDCs. Such a VO can be used to achieve the necessary scalability and availability.

13. Monitoring

The customer wants to monitor his/her application running on a remote data center.

14. Load balancing

The Grid monitors the job performance and adjusts allocated resources to match the load and fairly distributes end users' requests to all the resources.

For the scenario "Disaster recovery," the following functions are required in addition to the above:

15. Disaster Recovery

In case of the data center becoming unavailable due to a disaster such as an earthquake or fire, the remote backup data center takes over the application.

For the scenario described "Global load balancing," no additional function is required.

2.6 OGSA platform services utilization

The following services are necessary to provide functions in the previous section.

1. Name resolution and discovery service

This service is used for the Grid as discovery functionality.

2. Security service

This service is necessary for OGSA AAA functionality. Resource access control also needs the security service.

3. Reservation service

This service is used for advance reservation.

4. **Brokering service** This service is used for resource brokering.

5. Data management service

This service is used for data sharing within a data center and across them. It is also used for disaster recovery.

6. Provisioning and resource management service

This service is used for provisioning and also for creating a VO on a remote site.

7. **Scheduling service**

This service is used for priority job scheduling.

8. Metering and accounting service

This service is used for metering and accounting.

9. **Fault handling service**

   This service is used for fault handling. It is a part of autonomous management. In case of disaster recovery, affected IT business activities are relocated to other data center(s).

10. **Policy service** [6]

    This service is used for policy-related functionality.

11. **Monitoring service**

    This service is used for monitoring functionality.

12. **Deployment service**

    This service is used for provisioning functionality.

2.7 Security considerations

Each Commercial IT system (corresponding to a VO) should be securely isolated from each other since competing companies may be hosted in the same data center (RO). Before starting commercial systems, VOs should be divided using Virtual LAN or equivalent technology. When workload increases, IT resources (e.g. servers) will be reallocated to another system by rearranging the network configuration but no information should leak out.

WS-security is the starting point and some extensions may be necessary for the Commercial Grid System.

A VO may sit in a single data center or across multiple data centers. For disaster recovery and wide area load balancing, VOs should use multiple data centers.

2.8 Performance considerations

In contrast to the Science Grid, execution speed is not the highest priority requirement for the Commercial Grid. Instead, several Quality of Service matrixes should be considered. A best effort scheme cannot satisfy the Commercial Grid requirements. Since each job request should complete by the specified date and time, deadline scheduling by means of advanced resource reservation is the base-line assumption. Typically, jobs are expected to run for a certain predefined period and provide a certain level of performance.

To avoid the Internet spike problem, adaptive resource allocation (i.e., provisioning) enables scalability of the requests throughput.

Each IT system administrator expresses their requirements in a Service Level Agreement (SLA). Based on the SLA, each job demands additional resources under heavy load or substitute resources when a failure occurs. In case all requests cannot be satisfied, low priority ones, based on SLA, are rejected.

2.9 Use case situation analysis

Several cutting-edge technologies[7] [8] and products[9] already in the market attempt to solve one or more issues described above. Such attempts take a proprietary approach and have limited scope. OGSA, however, is an open, extensible, and comprehensive architecture, which can be used to address these problems.

[6] The explanation of policy service in [1] is very vague and is not clear what it is.

[7] Océano Project, IBM. www.research.ibm.com/oceanoproject

[8] N1, Sun Microsystems. wwws.sun.com/software/solutions/n1

[9] Jareva, http://www.jareva.com

We are now in research phase. After research completion, we would like to prototype OGSA-based CGS.

2.10 References

1. Foster, I and Gannon, D. The Open Grid Services Architecture Platform, 2003. www-unix.gridforum.org/mail_archive/ogsa-wg/doc00016.doc
2. Kishimoto, H., Savva, A., Snelling, D. OGSA Fundamental Services: Requirements for Commercial GRID Systems, OGSA-WG document, 14 October 2002 www-unix.gridforum.org/mail_archive/ogsa-wg/pdf00002.pdf

3 Severe Storm Modeling

3.1 Summary

A consortium of meteorologists and environmental modelers are attempting to build a Grid to enable them to accurately predict the exact location of severe storms such as tornadoes based on a combination of real-time wide area weather instrumentation and large-scale simulation coupled with data modeling. This is an extremely difficult problem and it is far beyond the current capabilities of storm simulation. Currently the meteorologists can only say that conditions for severe storms are favorable and issue warnings based on actual weather observations. Given the sighting of a storm, they can predict possible tracks, but given current compute and data analysis capabilities at their disposal, they cannot predict that a storm will appear at a specific location with any accuracy.

3.2 Customers

The primary customers are the meteorologists. The must actually use the Grid resources. This virtual organization is widely distributed and often mobile. A secondary set of customers are the emergency management people, disaster recovery teams and the mass media.

3.3 Scenarios

The scenario is roughly as follows. Instrument data streams from Doppler radar, satellite imaging, and ground-based sensors such as pressure, temperature and humidity detectors, are constantly monitored by data mining agents looking for dangerous patterns. When one is detected, VO members are notified and a large number of simulations are launched automatically. Data mining tools are configured to scan the output of the simulations and compare the results against the evolving data stream from the instruments. Data archives are searched for similar patterns. Some of the instruments are automatically reconfigured to refine the data streams.

As the storm evolves additional simulations are launched to refine the resolution of the predictions. Once a significant event is detected, humans monitor the entire process and aide in the process by steering some of the simulations. (The simulations generate output files which can be visualized as animations.) Other individuals on the ground are entering more data from mobile devices. The authorities and media are notified of the predictions.

This scenario is not yet possible because the Grid infrastructure is not yet in place. At the present time, many of the various components exist, but they are not all integrated. The current activity for this group is collaboration on testing the simulation and data mining and integrating the simulations with the data streams.

3.4 Involved resources

The primary resources involved include

- The sensor network courtesy of several agencies.
- The data archives of past storm activity and instrument readings
- The compute resources including the Teragrid resources

The services to be delivered:

- An integrated grid allowing VO members access to the simulation and data mining tools, the data archives and the sensor network tools.

- Eventually an automated, autonomic Grid of services that carry out the scenario described above.

3.5 Functional requirements for OGSA platform

3.5.1 Basic Functions

Discovery and brokering: Very large number of simulations and coupled data mining tasks are dynamically invoked when the weather turns bad. This requires discovery of resources and brokering to find resources of different sizes.
Data sharing: Very large databases of weather history (including radar data and other ground and space based data) must be accessed constantly. This information is distributed over hundreds of different databases. The evolving real-time weather is tracked against the historical information by data mining services and used to control the boundary conditions on the simulations.
Virtual organizations: Who has access to what parts of the instrument, data, compute resources is very important.
Monitoring: The large simulations must be monitored constantly to make sure they have the compute resources to continue. The entire grid of instruments and compute/data grid must be constantly monitored.
Policy: Policies control which members of the VO have access to the databases, instruments and the simulations. Policy also defines who must be notified when a severe storm is predicted. The notification process is automatically executed.

3.5.2 Security Functions

Multiple security infrastructures: Security controls who can control the on-line instruments.
Authentication, Authorization, and Accounting: These are all essential for management of the individuals in the VO and establishing their privileges.
Instantiate new services: Many of the services are simulation and data mining transient services. These must be instantiated on-the-fly by agents that are monitoring the data.

3.5.3 Resource Management Functions

Advanced Reservation: This is required for many of the scheduled data analysis tasks. However, the most important tasks have to be scheduled dynamically.
Scheduling: Dynamic scheduling is an essential component of this scenario. Compute resources must be provisioned on-demand to satisfy the need to complete a forecast on time.
Load balancing: If one resource becomes overloaded with simulation and data mining tasks, a new compute engine may be needed and the load can be balanced.
Notification/Messaging: Notification and messaging are critical in this very dynamic scenario. It is completely event driven.
Logging: Logging is required to understand what happed in the last "storm" so performance can be optimized later.
Workflow management: The workflow is very dynamic and is event driven.
System Properties
Fault tolerance: Better than real-time prediction requires extreme fault tolerance. The grid cannot go down while a severe storm is being tracked.
Disaster Recovery: Must be very fast. This may require that all computations be mirrored and very distributed.
The self-healing capabilities: The entire analysis/simulation/prediction scenario must be able to correct for its own errors.

3.6 OGSA platform services utilization

Required Services:

Name Resolution and Discovery: Severe store modeling must be able to discover data resources and data catalogs from metadata descriptions. This is part of discovery.

Service Domains: Collections of services need to be carefully coordinated. Resource brokers must assure compute and data storage resources. Network bandwidth must be available for on-time simulation and analysis. So these different types of brokers must be carefully coordinated

Security: Authentication is required by all members of the VO. However careful authorization polices with govern who has access to specific resources such as data or instruments. For example, not every VO member can be allowed to control an instrument.

Policy: Policy issues primarily involve access to instruments. Under what conditions can a radar be re-deployed? Also, policies will determine when a particular running system of services will be allowed to preempt resources for what "it perceives" as a critical need for public safety.

Data Management: Datagrid services: metadata catalogs, directory and index services, grid-wide access to data archives, virtual data management.

Messaging, Queuing and Logging: Grid-wide monitoring is needed by the resource brokers in order to provision the needed resources on time. Messaging and event systems are needed because of the very dynamic "demand driven" nature of the application workflow. Logging services are needed to understand what went wrong.

Events: Events are an essential component of this use-case. Monitors are constantly scanning instrument data streams looking for possible storm conditions. As they are found, event and message (pub/sub) systems will trigger the workflow scenarios essential to start the simulations and other data mining applications.

Metering and Accounting: Resource use costs money. Therefore billing has to be done and information required to do that has to be supplied.

Service Orchestration: Workflow engines have to orchestrate the coupled simulation /datamining/visualization tasks. The workflow has a very dynamic nature. External events, such as weather condition changes, can alter the flow of work. There are also time constraints on the work. If predictions are not completed on time, more resources may need to be allocated.

Administration: Software deployment is a serious administration issue.

Provisioning and Resource Management: Resource requirements change on a very dynamic basis. In the case of emergencies it must be possible to provision very large amounts of compute, bandwidth and data resources.

Reservation Services: yes. See provisioning and resource management.

Brokering and Scheduling Services: Compute and data resource brokering services are needed. Scheduling and co-scheduling services will be needed.

Fault Handling Services: Faults must be dealt with via system redundancy if better-than-real-time predictions are to be made.

Monitoring Services: Grid-wide monitoring, messaging, event systems and logging services.

3.7 Security considerations

The most serious security consideration is the case when an unauthorized user is given access to the instrument controls. This can cause substantial damage to the instruments if they are incorrectly used.

3.8 Performance considerations

Performance is an extremely critical component of this use case. Because the storm predictions must be made at better than real-time, it may be necessary to allocate huge amounts of computing and network bandwidth resources on-the-fly. A single storm may require 100 teraflops of dedicated performance over a period of several hours. This is currently not possible.

3.9 Use case situation analysis

None of the required services are in place at the present time. However, the instrument and data networks are there and there are many early ad-hoc experiments.

3.10 References

"A Modeling Environment for Atmospheric Discovery", the NCSA MEAD Expedition, see: http://www.ncsa.uiuc.edu/expeditions/MEAD

4 Online Media and Entertainment

4.1 Summary

To deliver an entertainment experience, several actors form a VO for this purpose. In a first step we want to focus on the following roles of actors:

- A consumer who consumes the entertainment content
- A service provider that hosts the entertainment content
- A publisher that offers the entertainment content
- A developer that consumes the entertainment Content.

Each roles may be consists of multiple companies and the entertainment content consists of many different forms (e.g. move on demand or online games) with different hosting capacity demands and lifecycle. Therefore one of the primary focuses of this use case is to facilitate the ability to dynamically manage resources based on workload demands and current system configuration. During the lifetime of an entertainment content the actors involved in the delivery of the content may change. During the lifetime of a company the entertainment contents it has to deal with may also change. Therefore the other primary focus of this use case is to provide standard interfaces to allow dynamic and open collaboration.

4.2 Customer and their need

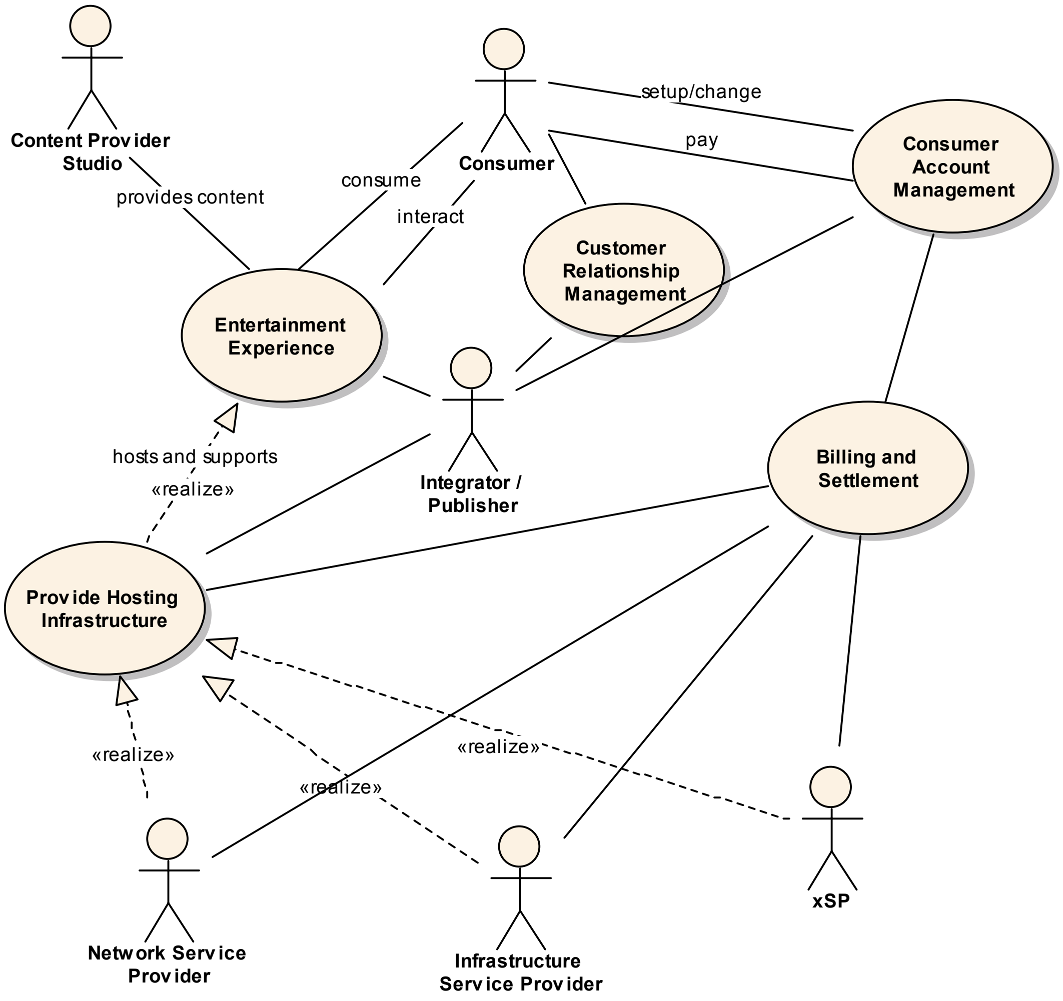
There are two main categories of entertainment experiences with each having unique requirements on the infrastructure that delivers it: consumption and interaction. Consumption of content (e.g. video on demand) does not require a lot of user interaction. Other contents, such as online games, require a lot of user interaction and it is very important to guarantee response times for these contents.

Online entertainment has seen a great adoption over the last couple of months. However, it is still in it's infancy in the areas of content, *business models and infrastructure*. With more online content available, differentiation from competitors will become more important. New commercial opportunities will emerge, for examples usage-based pricing or subscription models for premier consumer experience. Commercial transaction will be tied to entertainment or even inherent to the end user experience.

Because this is a new area, content developers *lack competency in programming for a distributed network.* There is no standard architecture or even best practice for how the back-end datacenters are used to deliver the contents. The most common practice today is to design one *stovepipe solution* for each game title, and *manage each solution separately*. Consequently, infrastructure and components deployed for each game are *not reusable*. Furthermore, these stovepipe solutions are designed with a particular level of workload assumed (e.g. 10,000 concurrent users), and scaling beyond this initially assumed workload requires major redesign. As a result, today's datacenters are either over provisioned, or overstressed to the point that service outage does occur. Finally, to make things worse, when a game is first designed, there is no way to tell how long the lifetime of the game is going to be. That is, the datacenters for these games may only be needed for only a few days (for a beta-test environment) or a few years (e.g. Everquest).

4.3 Scenario

In this scenario, there are 4 actors: consumers, service providers, publishers, and developers. A consumer, for example a game player, will access a portal and authenticate as a known identity. With this authorization he is then able interact with his account or consume an offered entertainment experience, e.g. play an online game. There may be several providers working in concert with each other. For example a network service provider that offer bandwidth, a hosting capacity provider who provides server and storage resources, and application service providers that offer common services like online game engines, standard customer relationship management and helpdesk applications or billing applications. The content provider or studio provides the media content, artwork and game play that the consumer will experience. The integrator or publisher ties the offering together and exposes it to the consumer. The figure below shows some simple interaction between these actors. The interactions between actors may change, and the entertainment content may change as well, therefore it is a key requirement to be able to autonomically manage resource allocation as well as enabling dynamic discovery and interaction of provided infrastructure and services.



The following table lists the main behaviors of each of the actors.

Consumer	Publisher	Studio	xSP
Sign up for account (with xSP)			Create Account
	Create a user subscription offer		
		Purchase and subscribe to hosting environments	Create a business offer for publishers (environment on demand)
			Provide subscription to game environment (includes reserve/scheduling and purchase)
	Delete a user subscription offer		Delete offering
Subscribe to contents/game(s) (with Publisher)	Create Authorization		Retrieve Authorization information
Authorization/authentication			Authorization/authentication
Find Content			Publish available contents
Create a M&E session			
Retrieve / use content			Create the On Demand hosting environment (provisioning, failover, workload management)
			Monitor Resources
			Add a physical resource
			Add new functionality/service
			Upgrade functions / services
			Delete an environment on demand offer
			Delete a physical resource from pool of servers
			Delete resources / services
			Load balancing
			Error capture, Problem Determination, Failover, and Recovery
	Define metering requirements		Meter usage
Apply a client patch/PTF			
			E-Commerce Integration
	Generate billing record based on billing and rating packages		Generate billing record based on billing and rating packages
	Bill player for usage (monthly, per hour, etc)		Bill publisher for usage/footprint

4.4 Resources and Services

The datacenter of online entertainment consists of at least the following components in a potentially distributed environment.

- Distributed Server
- Networked storage
- Secure network (including multiple levels of firewalls)
- Player Consoles

The online entertainment business includes at least the following functions:

- Security services (authentication / authorization, identity mapping, etc.)
- Financial services (billing, rating, accounting, etc.)
- Contracting / settlement services
- Customer relations services (logging and data mining of user behaviors)
- Management service (capacity management, workload management)
- Media / Entertainment specific services (e.g. multimodal input)

To solve the problem identified in section 2, the infrastructure hosting the online entertainment environment has to:

- Allow dynamic composition of standard pluggable components (e.g. billing service, customer relations service)
- Be secure and trusted.
- On Demand capacity (autonomic scalability according to workload), aggregation / selection of new services, integration with other companies that has needed competencies.
- There are currently major trust barriers in the online gaming industry, where publishers are very reluctant to share resources / components. To overcome this trust barrier, the components must be based on industry standard interfaces, and must be dynamically replaceable (i.e. the flexibility to choose components from a wide selection of providers).
- enable new commercial business models
- apply to needs of online game applications

More specific functional requirements, illustrated by specific examples, are listed in the sections below.

4.5 Functional Requirements

4.5.1 Discovery

OGSA services must be *discoverable at both runtime and setup time.* For example, a game developer needs to discover a set of rendering engines and choose to use a particular one based on the end user's screen resolution and connection bandwidth.
OGSA discovery must support *masking*; more specifically, render some services undiscoverable based on, amongst other things, a user's authorization and service level. There are different trust levels between companies. A company may want to expose all components of its software stack to a company that has a joint development agreement in place, but hide these components from other companies.

4.5.2 Instantiate new services

New service instances may need to be instantiated. For example, when an additional 2000 players joins an online game, a new game server needs to be provisioned to host these additional players. To provision the new server, the necessary services needs to be instantiated, and there are two aspects to this instantiation: deployment and scheduling/dispatching. Deployment involves transporting the necessary file / data to the server. An example of scheduling / dispatching may involve 1) reserving server resources for a period of time (e.g. reserve 2 hours to run AI logic) 2) determining the order of execution and whether the reservation can be met, and 3) dispatching the appropriate process when the scheduled time arrives.

4.5.3 Service Level Management

On of the biggest service level to be managed for online entertainment world is response time. For example, guarantee 50 ms response time for first person game, and 100ms for RPG.

4.5.4 Metering and Accounting

Resource usage needs to be logged with respect to each consumer and each provider. This information will be used to charge the consumers based on their usage, as well as used for cost analysis by the providers to determine the pricing.

4.5.5 Monitoring

The resource or service owners need to surface certain states so that the user of those resources or services may manage the usage using that state information.

4.5.6 Policy

There may be policies at every level of the infrastructure from the low level policies that govern how the resources are monitored and managed to high level policies that govern how business process such as billing are managed. High-level policies are sometimes decomposable into lower-level policies.

4.5.7 Grouping / Aggregation of Services -- based on policy and functional requirements

Taking on-line games as an example, the game developers lack competency in many areas such as network programming, rating and billing, eCommerce integration, etc. Therefore, composing services using existing services is a core requirement. There are two main types of composition techniques needed by the online gaming developers: selection and aggregation. *Selection* involves choosing to use a particular service amongst many services with the same operational interface (e.g. select the *fastest* MP3 encoder.) Aggregation involves *orchestrating* a functional flow (workflow) between services. For example, the output of accounting service is fed into the rating service to produce billing records. One other basic function required for aggregation services is to *transform* the syntax and/or semantics of data or interfaces.

4.5.8 Security

In such a flexible environment, resources will over time be used for multiple content titles. Therefore trust has to be built on the side of the content providers that such a dynamic environment will not interfere with the goal of consistent user experience. Proper isolation between content offerings also has to be ensured. This level of isolation has to be ensured by the security of the infrastructure.

In addition, several securities related services are required:

- Single sign-on needs to be supported. A player may traverse several organizations in the M&E environment. For example, a player of Everquest may buy an Everquest character on e-bay and pay for it via his pay-pal account. To support single sign-on a game developer may want to use a 3rd party authentication and authorization service, identification mapping service, etc.
- Digital rights management and key management.
- Intrusion detection and protection

4.5.9 Certification

A trusted party certifies that a particular service has certain semantic behavior. For examples, a company will only use e-commerce services certified by yahoo shopping.

4.5.10 Lifecycle / Change management

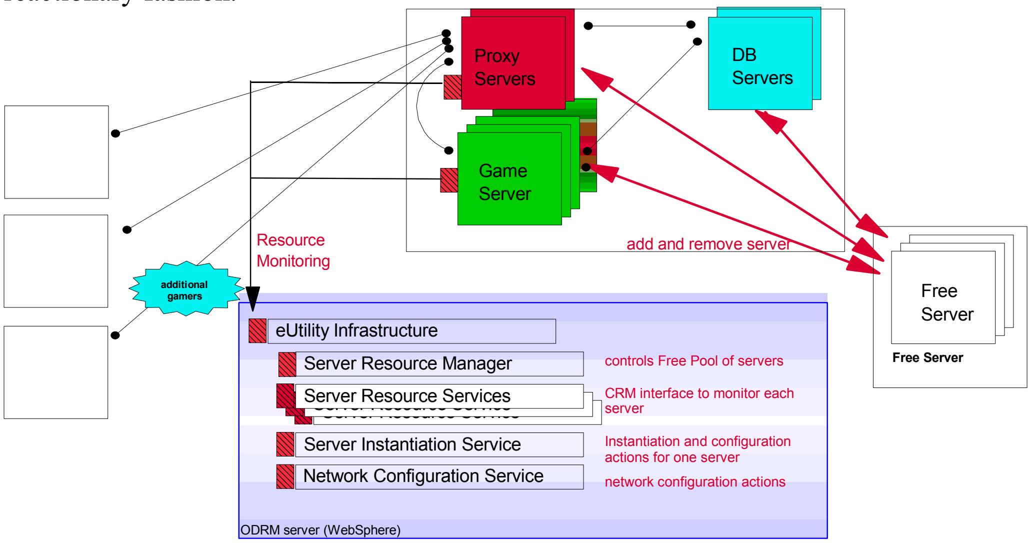
Upgrade or retire services with minimal impact to deployed and running services. This could be accomplished by a workflow which provisions the required services, and dynamically modifies the current running environment by changing its selection rules and / or workflows.

4.5.11 Failure Management

OGSI soft state management could be one way to implement a heart beating function. Resource instrumentation can provide additional information about how well resources are functioning. Logging service is needed to keep track of resource's history of performance and is necessary for error capture and trigger recovery actions. For example, when a game server's performance is degraded because of a software problem, apply patch.

4.5.12 Provisioning Management

Take online gaming as an example of the M&E industry. On-line games' workloads are very close to uniform sinusoidal waves, but typical server farms are still only about 20% utilized. It is ideal for the providers of the data centers to not over provision for the peak workload, but instead, use just enough capacity to meet the required service level agreements in both a predictive and a reactionary fashion.



4.5.13 Workload Management

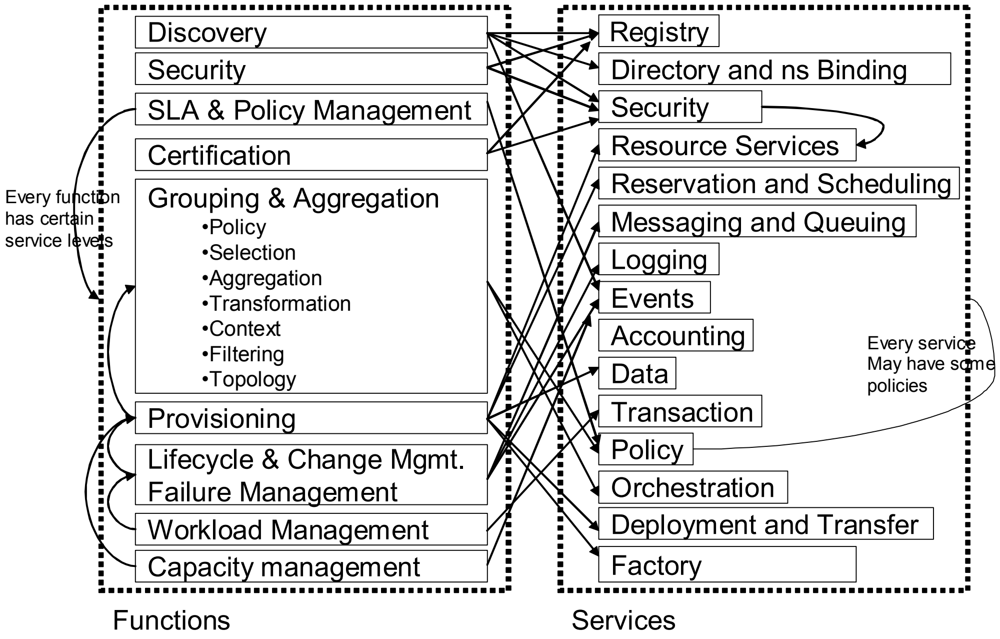
Taking online games as an example, the amount of workload is a direct result of how many concurrent players are being hosted on a game server. If the game server A is responsible for a 20 square mile area in the game world, and a battle occurred in that area, many players will rush to that area, causing workload on that server to increase. As players enter that area and leave other areas, other servers' workload will decrease. So, when the workload of server A gets above certain threshold, a load balancing routine needs to be triggered to rebalance the resources (i.e. servers). That is, redistribute workloads across servers with idle capacity.

4.5.14 Application Specific (e.g. multimodal input) services

Additional domain specific services may be needed; for example, a voice recognition engine.

4.6 OGSA Service Mapping

Possible OGSA Porttype / Services:



Discovery
Security
SLA & Policy Management
Certification
Grouping & Aggregation
- Policy
- Selection
- Aggregation
- Transformation
- Context
- Filtering
- Topology

Provisioning
Lifecycle & Change Mgmt.
Failure Management
Workload Management
Capacity management

Every function has certain service levels

Registry
Directory and ns Binding
Security
Resource Services
Reservation and Scheduling
Messaging and Queuing
Logging
Events
Accounting
Data
Transaction
Policy
Orchestration
Deployment and Transfer
Factory

Every service May have some policies

Functions

Services

4.6.1 Security Considerations

Each consumer, service provider, developer and publisher must have its own security identity and context (e.g. relationships with other entities). All security functions traditional in the enterprise environment must be addressed including privacy and non-repudiation.

4.6.2 Performance Considerations

The backend server infrastructure has to be able to scale driven by increasing concurrent number of consumers and amount of content. Another aspect of scalability is the number of content pieces or game titles that will be served by a single datacenter. New titles will also require more compute, network and storage resources per player.

4.6.3 Situation Analysis

Several cutting-edge technologies and products already in the market attempt to solve one or more issues described above. Such attempts take a proprietary approach and have limited scope. The OGSA, however, is an open, extensible, and comprehensive architecture, which can be used to address these problems.
We are now in proof of concept phase, after which, we would like to prototype OGSA base Commercial GRID system.

4.7 Reference

5 National Fusion Collaboratory

5.1 Summary

The National Fusion Collaboratory (NFC) project [References: 2] defines a virtual organization devoted to fusion research and addresses the need of software developed and executed by this community. Up to now, the developers would typically port their software to a standard set of platforms and the community users would then install and use this software on their machines. This process was found to be complex from the viewpoint of the provider as well as user. The user after going through the usually complex process of installing the binaries and its dependencies would then have to contribute to maintaining that software whenever a new version comes out. This process is made especially difficult by the fact that scientific codes are typically developed and refined over decades and result in very complex systems which need to be updated frequently in order to reflect the latest improvements in modeling and simulation techniques. From the provider's point of view, the necessity of supporting the software on even a limited set of platforms can require significant cost and effort. In addition, maintaining and debugging community software on an unfamiliar platform can mean a significant amount of effort in reproducing, let alone fixing, a problem.

Due to these problems, the fusion community recently decided to adopt the application service provider (ASP) model, also known as the "network services model". In the "network services" model, software as well as a set of familiar platforms is provided or contracted by a service provider and made accessible remotely to clients. The service provider undertakes not only to maintain a reasonable set of versions of the software, but also to debug and otherwise manage client execution runs to ensure that they achieve their objective. This might include executing the software as efficiently as possible, executing it within a certain time bound, producing results of a certain accuracy (see next section for details). The clients specify those objectives and execute the codes remotely thus avoiding maintenance costs. This sharing paradigm is new to the Fusion community, but is rapidly gaining acceptance as it encourages sharing of software and hardware resources and frees the researcher from needing to know about software implementation details and allowing a sharper focus on the physics.

5.2 Customers

The customers of this use case are fusion scientists. Service providers defined above seek to reduce maintenance costs by providing a service on a familiar set of platforms, while service clients seek to obtain remote execution of a software satisfying certain objectives, specifically capable of executing within certain time bounds during fusion experiments. Two principal issues arise in this environment - issues of trust and issues of control.

Issues of trust address questions such as: will my software execution run get priority when I need it? How do I enter into contract with software/hardware resource provider? What guarantees do I have that this contract will be observed? And, on the provider's side - how can I ensure that my deployment is secure and yet deal with a dynamically changing community of users?

The issues of control deal with questions like - how do I provide reliable execution in this environment? How can I meet client's demands? All of these issues need to be addressed in a wide-area deployment which is national and eventually international comprising hundreds and potentially thousands of users at a later stage.

Below we summarize in detail the needs of the clients as well as providers.

QoS-based execution during fusion experiments: Magnetic fusion experiments operate in a pulsed mode producing plasmas of up to 10 seconds duration every 15 to 20 minutes, with multiple pulses per experiment. Decisions for changes to the next plasma pulse are made by analyzing measurements from the previous plasma pulse (hundreds of megabytes of data) within roughly 15 minutes between pulses. This mode of operation could be made more efficient by the ability to do more analysis and simulation in a short time using software running on remote resources *only if* their execution time could be guaranteed. Given the present capabilities, the decision to include new software in the "between pulse" analysis usually involves buying a new cluster that will be run on-site and dedicated during the experiment. Obviously this mode of operation does not scale in the long run. The ability to run software on remote resources would be helpful, on the condition that end-to-end quality of service (QoS) guaranteeing the execution within certain time bounds could be provided. For example, end-to-end quality of service should combine input and output data transfer and execution time and ensure execution of this QoS-based workflow in such a way as to meet the user's overall QoS requirement.

Availability contract: Like in many other scientific communities, much of the work in the Fusion community is driven by the need to make results available in time for major conferences. Although the current deployment has not yet been found lacking in this respect, we anticipate that resource utilization before such events will grow to the point where some user's requests will not be fulfilled due to high demand. The resolution of this problem could be provided by a contract mechanism whereby the user contracts for the availability of a service ahead of time, and claims it when the need arises.

Usage policies: Both of the client needs described above require mechanisms for usage policy specification and enforcement on the part of service/resource provider as well as the virtual organization. The service provider for example has the need to assert who (which groups or users) have the right to run certain software, the resources they can use, the availability contracts they can enter into, the service execution management etc. Such usage policies also have to be suitably enforced by the underlying resource management system.

Flexible delegation of rights: Providing seamless maintenance of a client's run requires flexible rights and delegation policies for the server. For example, if a run is found to experience an unexpected failure, the service provider may want to diagnose the run, debug and restart it. Since the run may involve access to secure databases, in order to perform these actions, the service provider will need to acquire rights that allow it to reproduce this usage pattern. Impersonating the client is not necessarily a reasonable option as that may give the service provider too many rights, and the client may be unwilling to do this.

Community accreditation: The clients would like to be able to use community services by getting accredited with the community rather than each individual service provider. For example, code execution on a hardware resource (which may not even be known to the client) should not be associated with the need to obtain an account on that resource. Instead, a mechanism is needed whereby it is sufficient for the client to present community credentials in order to initiate the run.

5.3 Scenarios

In the experimental scenario described above, a scientist at one of the NFC sites (a client site) needs to remotely run code installed and maintained at another NFC site (a service provider site) during an experiment within time bound *T* (typically on the order of 10 minutes). For a very simple execution, the following would be available on the service provider's side: a script that will download experimental data for the application input once that data becomes available, a suitable "short-running" configuration of an application, capable of executing in less than *T* (some applications may be available in multiple configurations reflecting accuracy/time trade-offs), a script delivering results to the client, as well as an execution plan, or a workflow, describing the sequence of these actions and their QoS dependencies. To ensure that the code executes with the required QoS (in this case: within time *T*), the scientist at the client site enters into a contract with the application server and as a result is guaranteed code execution within *T* any time it is requested during the experimental availability window (typically a day). Since only a few such executions may be requested during that day, and the service provider resources have to be shared with other clients, it is essential that resource allocations are not overgenerous and that other software can share the resource with the time-critical application, getting preempted whenever the situation requires.

When the client claims the execution based on the contract, the service provider initiates and monitors the run, adaptively recovering from failure of specific actions if needed. Depending on the importance of the run the service provider could overprovision, or replicate the run.

This scenario can become more sophisticated depending on the service in question. It is essential that the execution time or other QoS aspects experienced by the client is end-to-end, in other words service provider accounts not only for application execution but also allows for database access, data transfer, and other activities. It is important to note that data availability before transfer time (replication) cannot be leveraged in this case as it becomes available dynamically. Similarly, in national (and potentially international) deployment data transfer will become a significant factor which cannot currently be reliably managed. Also, it is important that the QoS-base execution is available to small fusion labs in small centers as well as large fusion labs in large centers.

Apart from the time, fusion codes can also require non time-critical mode of execution but one that provides accurate results, or the time requirement can be relaxed to complete by a certain deadline rather than in a specific amount of time. More details of the scenario are described in [References: 3].

5.4 Involved resources

The primary resources involved include

1. The hardware resource at service provider site; these can range from supercomputers to single workstations.

2. The machines running the client's sites.

3. Networks between Fusion sites (the service provider sites and the client site), they are widely distributed, potentially internationally distributed.

The services to be delivered primarily relate to service executions, and may involve experimental hardware services (e.g. experimental apparatus) in the future.

5.5 Functional requirements for OGSA platform

This use case uses the following OGSA functionalities as described in [1]:

1. **Discovery.** The clients need to discover network services before they are used. Service brokers need to discover hardware and software availability.
2. **Workflow management**. A fusion grid network service is a workflow of multiple components (remote execution, input and output data transfer, etc.).
3. **Scheduling of service tasks**. The service provider (or broker) acting on service provider's behalf needs to schedule resource in order to meet the execution constraints requested by the client. The scheduling can take the form of advance reservation.

4. **Disaster Recovery**. As the service provider (or broker acting on its behalf) strives to meet the client's end-to-end constraints, some degree of adaptation may have to be used to prevent failure.
5. **Brokering**. The service broker identifies software and platforms suitable for execution requested by the client.
6. **Load Balancing**. Some load balancing may be required to use service provider resource more efficiently.
7. **Fault Tolerance.** A reliable solution is needed in order to provide the time-critical execution capability.
8. **Transport Management**. Reliable transport management is essential to obtain the end-to-end QoS required by this application.
9. **Legacy Application Management**. Realizing the Grid potential to deal with legacy issues was the one of the foremost motivation for this project.
10. **Services Facilitating Brokering**. This capability is essential for the service broker to compose and later execute a workflow meeting the requested constraints.
11. **Application and Network-level Firewalls**. This is a long-standing problem in the fusion use case. It is made particularly difficult by the many different policies we are dealing with and particularly harsh restrictions at international sites.
12. **Agreement-based interaction**. This project requires agreement-based interaction capable of specifying and enacting agreements between clients and service providers (not necessarily human) and then composing those agreements into higher-level end-user structures.
13. **Authorization and usage policies**. We also require use policy specification and enforcement mechanisms as described above.

5.6 OGSA platform services utilization

The following services are necessary to provide functions in the previous section.

1. Name resolution and discovery service
2. Security service
3. Provisioning and resource management service
4. Metering and accounting service
5. Policy service
6. Messaging and logging
7. Monitoring service
8. Metering and Accounting
9. Administration
10. Service Orchestration

5.7 Security Considerations

The server sites need the ability to provide authorization on the usage of certain software (or application services) as well as on the usage of resources. The VO-specific authorization policies need to be maintained centrally, while resource-specific policies need to be maintained by resource owners.

In addition, application service providers need to be able to assume a subset of user's rights needed to debug an application that has gone astray. This is needed because applications access the experimental database based on the rights of the user that started the run. Frequently, the application provider is able to debug and resubmit the user's program in a manner transparent to the user.

5.8 Performance Considerations

The ability to deliver services in real-time is essential. Also important is the ability to satisfy other QoS constraints (application-specific notions of accuracy).

5.9 Use case situation analysis

Some of the required capabilities have already been provided by Globus as evidenced by the fact that fusion services are deployed and successfully used by the community. Currently research in enforcement issues, issues of agreement-based interaction, as well as scheduling and adaptive techniques that would support them are going on. Also required are changes in the security model and advances in overcoming deployment issues such as firewalls.

5.10 References

1. Foster., I. and Gannon, D. The Open Grid Services Architecture Platform. www-unix.gridforum.org/mail_archive/ogsa-wg/doc00016.doc.
2. Keahey, K., Fredian, T., Peng, Q., Schissel, D.P., Thompson, M., Foster, I., Greenwald, M. and McCune, D. Computational Grids in Action: the National Fusion Collaboratory. Future Generation Computing Systems (to appear), 18 (8). 1005-1015.
3. Keahey, K. and Motawi, K. Taming of the Grid: Virtual Application Services, Argonne National Laboratory, Mathematics and Computer Science Division Technical Memorandum ANL/MCS-TM-262, 2003.

6 Service-Based Distributed Query Processing using OGSA and OGSA-DAI

6.1 Summary

A service-based distributed query processor supports the evaluation of queries expressed in a declarative language over one or more existing services. These services are likely to include database services, such as those provided by the OGSA-DAI project (www.ogsa-dai.org), but may also include other computational services. As such, a service-based distributed query processor supports service orchestration, and can be seen as complementary to other infrastructures for service orchestration, such as workflow languages. In a Grid setting, distributed query processing can benefit from the facility to discover and make use of computational resources on demand, based on the anticipated resource requirements of a request. A distributed query processor on the grid can itself be cast as a service, referred to here as a Grid Distributed Query Service (GDQS).

In principle, a GDQS can be used in any Grid application that must integrate and analyze structured data collections. Regardless of the application domain, there are several primary phases in a typical use case involving GDQS. Some of those phases are transparent to the user, whereas some require interaction with the user. All, however, imply particular requirements from the grid software infrastructure. Each phase will be examined in more detail in Section 1.3; below is a summary:

- **Factory discovery and service instance creation phase.** The user has to discover a GDQS factory by querying a Grid Data Service Registry (GDSR). It is the users' responsibility to have the knowledge of an appropriate registry and a reasonable search criterion. Once the factory is discovered an instance can be created.
- **Resource discovery phase**. The GDQS needs to obtain metadata about the computational capabilities of available grid nodes in order to be able to optimize and efficiently schedule a query plan. This phase is transparent to the user.
- **GDQS setup phase.** The User is required to prepare the GDQS instance for accessing multiple data sources and analysis services. This involves providing the factory handles and an appropriate configuration document for OGSA-DAI services that wrap the data sources being integrated, as well as providing the WSDL URLs of the services that are to be used for analysis. The GDQS uses this information to import the database schemas of the data sources and WSDL content of the services so that it can process (compile and optimize) the submitted query.
- **Query (request) submission phase**. The user is required to formulate a query in Object Query Language (OQL) and submit it to GDQS.
- **Query Execution and result delivery phase.** Once the query is submitted, the GDQS compiles, optimizes, schedules and executes the query utilizing the available computational resources on the grid by taking into account the information collected in the resource discovery and GDQS setup phases. The results are, then, delivered to the user subject in the interaction patterns allowed by the OGSA-DAI Grid Data Service (GDS) port type interaction semantics [ref to OGSA-DAI].

6.2 Customers

The potential users of SB-DQP can be both from commercial or scientific background. A fundamental characteristic of the usage pattern is the requirement to integrate data from distributed and heterogeneous resources with analysis capabilities provided as services. For example, distributed query processing is considered a relevant technology in bioinformatics, in which there are many distributed structured data stores, and in which an individual analysis often needs to access several of these stores and several analysis tools. In bioinformatics, there are several hundred important structured data stores (of very variable size) and many analysis tools applicable to data that can be extracted from these stores. Currently many bioinformaticians apply a sequence of disconnected (or largely manually connected) activities to achieve data and analysis integration. A declarative interface that uses a standard query language to combine such disconnected activities in an optimized way is of particular interest to the bioinformatics community.

A detailed scenario that illustrates the potential value of the GDQS for bioinformaticians is given in Section 1.3. The scenario provided illustrates the integration of data from two distributed data resources, the Gene Ontology (GO) database, the Genome Information Management System (GIMS) in combination with an analysis tool, namely BLAST.

6.3 Scenarios

The following OQL query is meant to provide a starting point for constructing a scenario that illustrates how a bioinformatician can interact with a GDQS causing it to pass through the phases introduced in Section 6.1. First the query is explained and then scenarios are provided that exemplify the

```
select p.proteinId, blast (p.sequence)
from p in protein, t in proteinTerm
where t.termId='GO:0008372' and
p.proteinId=t.proteinId
```

This query returns, for each protein annotated with the GO term 'GO:0008372' (i.e., unknown cellular component), those proteins that are similar to it. Assume that (as in [21]) the protein and proteinTerm extents are retrieved from two databases, respectively: the Genome Information Management System (GIMS) [img.cs.man.ac.uk/gims] and the Gene Ontology (GO) [www.geneontology.org], each running under (separate) MySQL relational database management systems. The query also calls the BLAST sequence similarity program [www.ncbi.nlm.nih.gov/BLAST/], which, given a protein sequence, returns a set of structures containing protein IDs and similarity scores. Note that the query is essentially a select-project-join query but retrieves data from two relational databases, and invokes an external application on the join results. A service-based approach to processing this query over a distributed environment allows the optimizer to choose from multiple providers (in the safe knowledge that most heterogeneities are encapsulated behind uniform interfaces), and to spawn multiple copies of an operator to exploit parallelism. In the example query, for instance, the optimizer can choose between different GO and GIMS databases, different BLAST services, and different nodes for evaluating the query sub-plans.

6.3.1 Service Discovery and Instance Creation

Figure 1 illustrates the interaction during the first phase. The first interaction in the figure refers to the fact that a GDQS factory registers itself to a GDSRegistry as part of its initialization. The client queries a Registry using `GridService::FindServiceData` operation to find an appropriate GDQS factory (GDSF) (interaction 2). The client then creates an instance of the GDQS using the OGSA factory port-type.

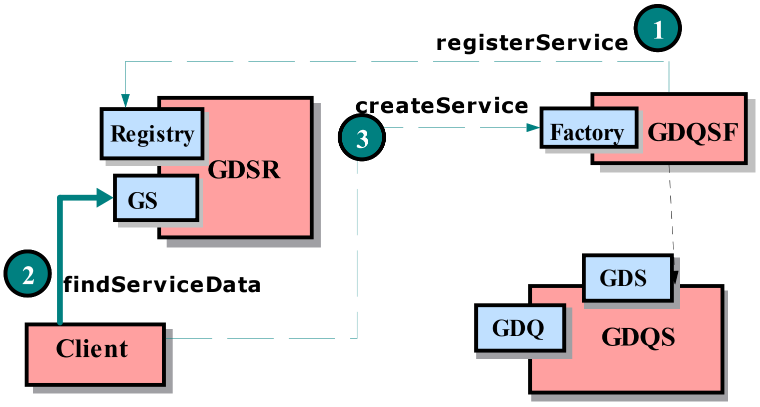


Figure 1 Service Discovery and GDQS instance Creation

6.3.2 Setting up the GDQS instance

It is necessary for the GDQS to collect database schema information of the data sources being integrated. Figure 2 illustrates interaction during the setup phase through which the GDQS acquires this information. The client discovers a GDS Factory for a particular data source (interaction 2) and passes the handle of this factory (GSH:GDSF) along with a configuration script obtained by querying the factory (interaction 3), to the GDQS instance via an import Schema call (interaction 4). It is also necessary to provide a configuration document to determine the type of the GDS being created. The client should be able to interrogate the GDS factory to find out the set of configurations supported, and choose the most convenient one. The GDQS instance, then, creates a GDS instance (GDS1) using the factory handle and the configuration document provided by the client (interaction 5), and obtains the database schema of the data source wrapped by that GDS (interaction 6).

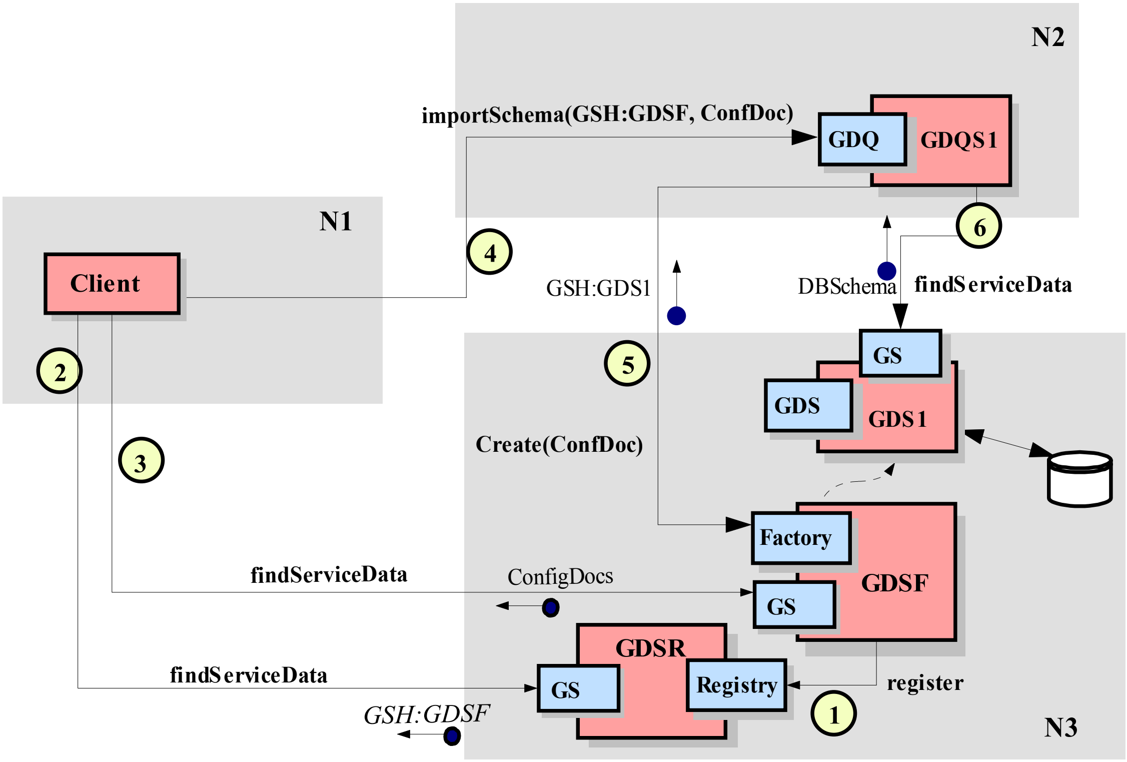


Figure 2 Importing Schema Information of Data Sources

6.3.3 Collecting Computational Resource Metadata

It is also important for the GDQS to collect sufficient data about the available computational resources on the Grid to enable the optimiser to schedule the distribution of the plan partitions as efficiently as possible.

Although the current OGSA reference implementation does not fully support this need, it does provide a high-level Index Service, to enable collecting, caching and aggregating of computational resource metadata. Figure 3 illustrates the service-based architecture that enables a GDQS to collect resource metadata from multiple nodes on the Grid. In this set-up, an index service collects dynamic information on the system it is deployed in using back-end information providers. The GDQS identifies a central index service as its server for caching and aggregating metadata, and causes (2) it to subscribe to other distributed index services. The remote index services send (3) notification messages at specified periods whose payload is resource metadata in a format determined by the back-end information provider. The GDQS can use (4) a findServiceData call to obtain the aggregated information as SDEs from its server.

Note that one would expect the index service hierarchy to have been set up as part of a virtual organisation's infrastructure, since the identification of Grid nodes that constitute the organisation's resource pool is beyond the operational scope of the GDQS.

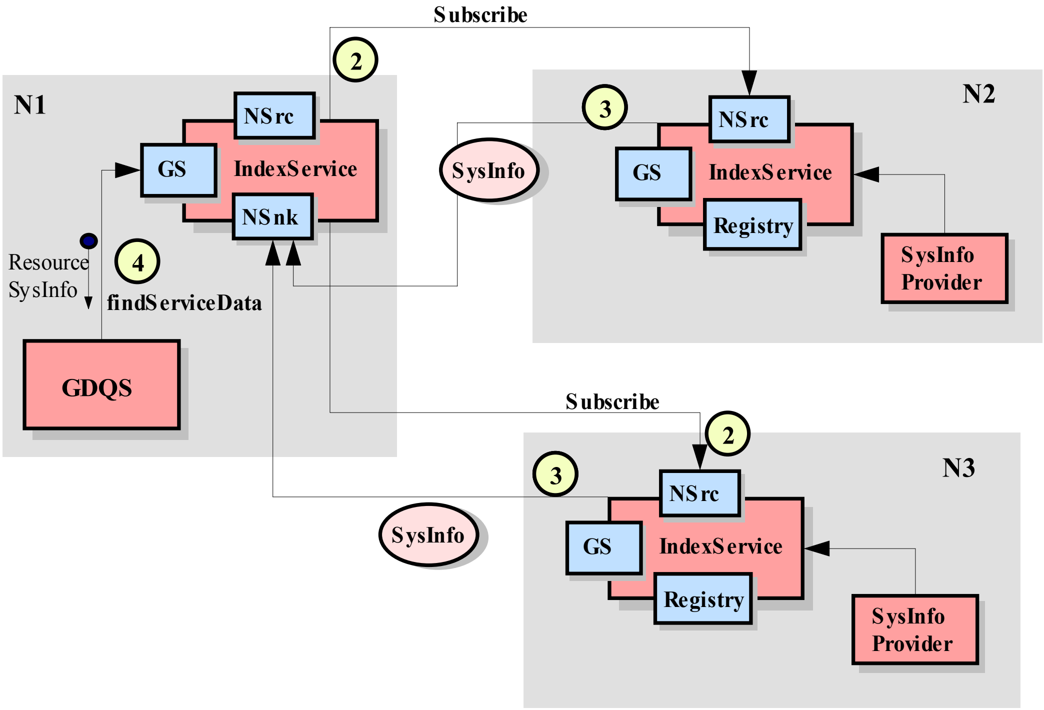


Figure 3 Acquiring Computational Resource Metadata

6.3.4 Query (Request) Submission

Most of the interactions (apart from the initial query submission) in this phase are inter-service interactions transparent to the user. Figure 4 illustrates those interactions. After importing the schemas of the participating data source, the client can submit queries (1) via the GDS port type using a perform call. Note that the format and semantics of query submission is compliant with that of OGSA-DAI framework. The submitted query is compiled and optimized into a distributed query execution plan. The GDQS, then creates a set of Grid Query Evaluator Services (GQES) for executing each query-sub plan (or partition) generated by the query optimizer on a different node on the grid. The scheduling of the GQES instances is also done in an optimized way based on the metadata collected. Once the GQES instances are created on their designated execution nodes (and these could be, potentially, anywhere in the Grid), the GDQS hands over to each (2) the plan partition assigned to it. This is what allows the DQP framework to benefit from (implicitly) parallel evaluation even as the uniform service-based interfaces hide most of the low-level complexity necessary to achieve this. Finally, (some of the) GQES instances interact (3) with other GDS instances to obtain data, after which the results start to propagate (4) across GQES instances and, eventually, back to the client via the GDT port type.

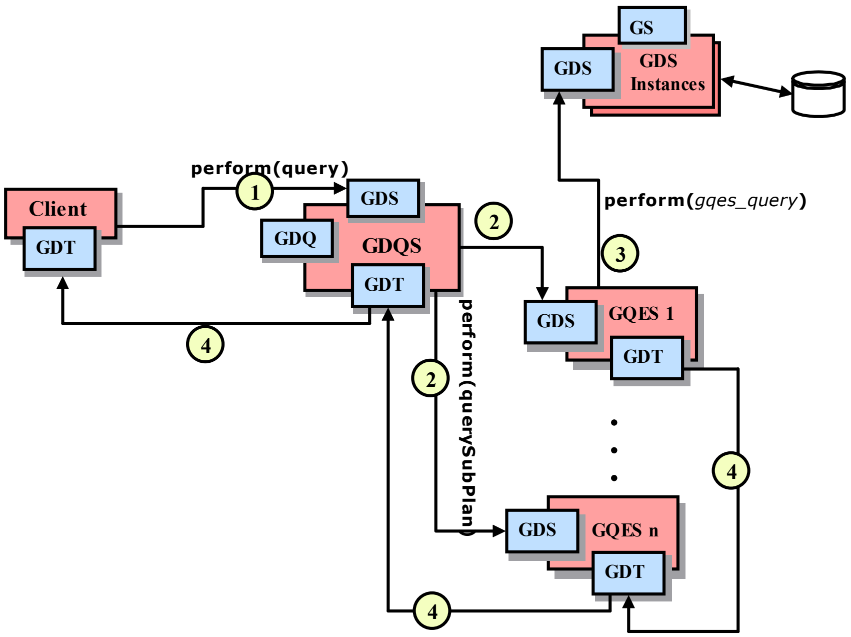


Figure 4: Query Execution - Overview

6.3.5 Query Execution and Result Delivery

For the example query given at the beginning of Section 6.3, the query submission gives rise to the Grid Service (GS) interaction diagram in Figure 5. The GQESs that scan stores, viz., N1 and N2, are instantiated in different hosts. Conditions at N2 (e.g., available memory) are such as to justify the GDQS having assigned the hash join to N2. For the BLAST operation call, the GDQS saw benefits in parallelizing it over two GQESs N3 and N4. The GDQS receives the request (1) and compiles it into the distributed query plan in Figure 7(d), each partition of which is assigned to one or more execution nodes. Each execution node corresponds to a GQES instance which is created by the GDQS (2). The GDQS then dispatches (3), as an XML document, each plan partition to its designated GQES instance. Upon receiving its plan partitions, each GQES instance initiates its evaluation. Query execution is a data flow computation using the iteration model, in which each operator implements fopen (), next (), close () interface. Data flows from the GQES instances that execute partitions containing operators whose semantics requires access to stores.

Within each GQES instance, the initialization procedure starts when an open () call reaches the topmost operator. This call propagates down the operator tree from parent to children at every level until it reaches the leaf operators. Then, interaction with other GDSs occurs. The handle for each such GDS will have been planted by the GDQS in the XML document passed to each GQES instance that needs it. For example, in node N2 (in Figure 5), when the stream of open() calls reaches the sequential scan operator, it causes the N2 GQES to interact with the GDS instance on N2, whereby data becomes ready to flow upwards from the protein extent in the GDS through which the GIMS database is accessed.

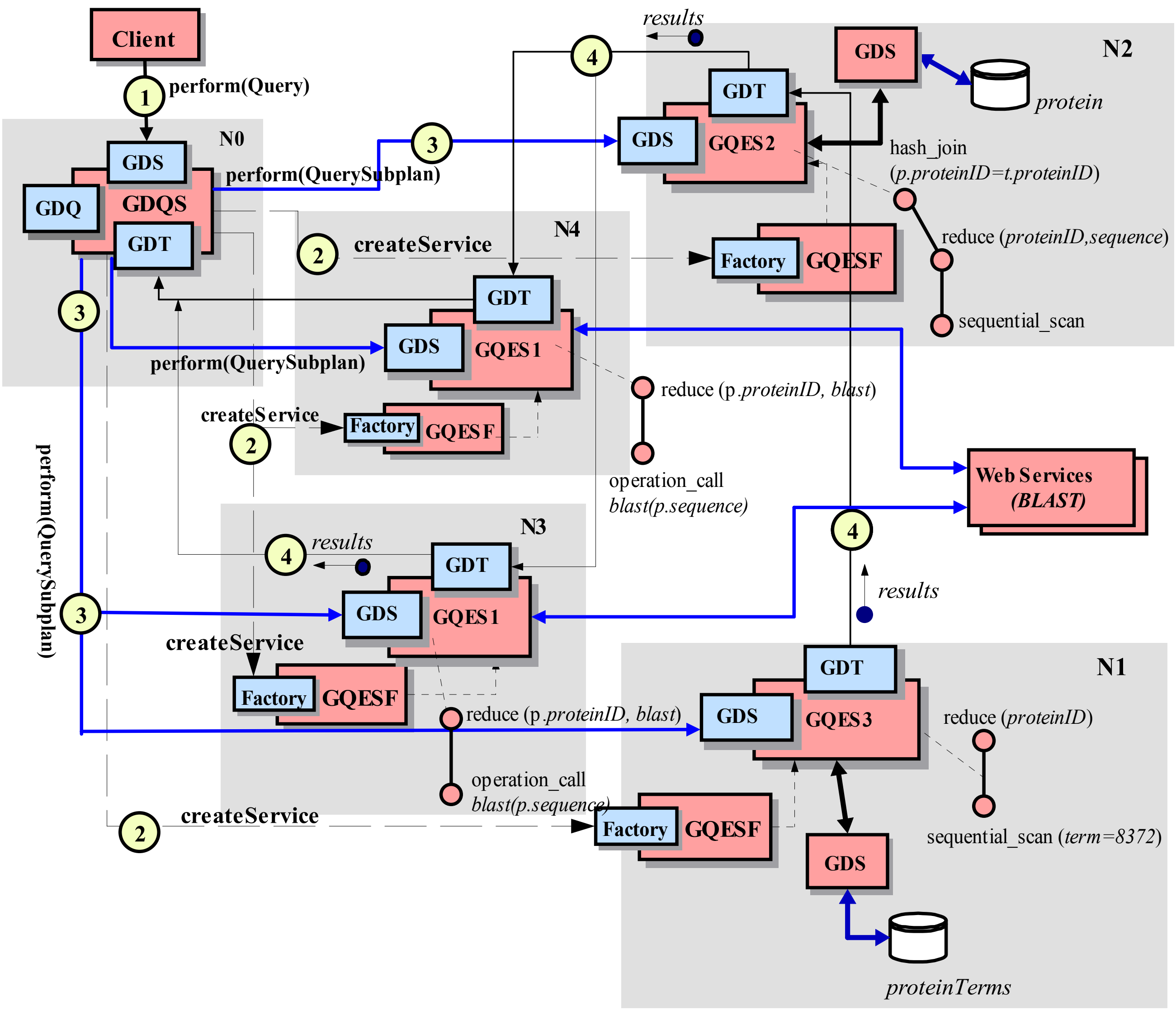


Figure 5: Query Execution and Result Delivery - Detailed

Note that many forms of disruptive heterogeneity in the data stores are encapsulated by the standard GDS interface. As such, SB-DQP exploits the power that the Grid metaphor embodies, viz., query evaluation is carried out over heterogeneous data and computational resources but the heterogeneity is encapsulated behind the universal GS interface, giving rise to consistent and uniform inter-service interaction semantics.

6.4 Involved resources

A GDQS can be expected to make use of computational resources for: (i) running query evaluator services, several of which may collaborate in the evaluation of a single query; (ii) moving data from primary sources to analysis tools or to evaluators that join or manipulate the data in a query; and (iii) holding intermediate results for performance or reliability. All such computational services need to be identified and allocated dynamically to support the specific needs of complex requests.

In terms of the services used by a GDQS, these are likely to include: (i) service registries, as service descriptions must be imported into a GDQS before queries are evaluated over them; (ii) structured data access services, as consistent access to structured stores is important for reducing set-up costs; and (iii) flexible transport services, for example supporting streaming of data and delivery to multiple sites in parallel.

6.5 Functional requirements for OGSA platform

- *Discovery and Brokering.* It is very important for SB-DQP to be able to discover available computational resources, Grid Data Services (GDS) and Analysis Services (AS). The discovery of the GDSs is needed for importing the database schemas of the data sources over which a query is to be formulated. Discovery of analyses services is needed to identify the type of operations and data types supported/required by those operations, so that they can be embedded in a query. The crucial requirement here is a uniform model that will enable both the SB-DQP clients (users) and the DQP service itself to discover and interpret the metadata about such services but also to relate them to the information about computational resources (hosting environments, machine capabilities such as CPU speed, available memory etc.).

- *Metering and accounting.* SB-DQP can potentially use many GDSs and other grid and web services. Each of these may have its own impact on the overall billing cost of the distributed query service. SB-DQP must be able to integrate into metering, accounting and billing mechanisms employed by other participating data sources and/or services and if possible choose from among the most convenient ones based on user preferences. This is only possible if such seamless integration is supported at the infrastructure level.

- *Data sharing and management.* Data sharing and management is fundamental to SB-DQP. It does this at two levels. At the lower level it relies on Grid Data Services for accessing data sources, and at a higher level it processes the data it obtains (joins, reduces, analyses etc) in a way that conforms to the principles of a data-flow architecture. It does not however, currently, address the problem of schema integration and consistency. SB-DQP would benefit from such data management facilities as semantic data model integration, transparent data caching and consistency management.

- *Monitoring.* SB-DQP requires monitoring in several contexts. First, it should monitor the progress of the services it orchestrates. Progress information has to be collected from the Evaluator services (GQESs), GDSs and analysis services. Second, since a query can potentially involve long running interactions (because of large amounts of data or network conditions) the SB-DQP should respond by re-allocating resources and re-scheduling evaluator services. This, in turn, requires monitoring of computational resources to collect dynamic information to aid in reaching a decision as to how to adapt to the changing conditions.

- *Multiple security infrastructures.* In most of the cases the distributed query will require access to multiple data resources access to which may be restricted by different security policies and infrastructures. It is essential for the SB-DQP to rely on infrastructure support for obtaining access permission to multiple resources on behalf of the client in a transparent way.

- *Optimization of resource usage.* SB-DQP uses a query optimizer (the Polar* system) which is responsible for generating an efficient execution plan for a declarative OQL query over distributed services (both data and computational, since OQL supports invocation of external functions). As such, SB-DQP offers system-supported optimization of declarative requests with implicit parallelism. In that respect

- *Transport Management.* As the SB-DQP executes the queries as a data flow computation, efficient data transport is of paramount importance. Shipping only XML data over SOAP/HTTP is not particularly convenient for data intensive applications. It is very desirable to have multiple transport protocols, including very efficient ones, to be available for inter-service interactions.

- *Fault tolerance and disaster recovery.* Fault tolerance is particularly important for long running queries that can potentially return large amounts of data.

6.6 OGSA platform services utilization

- *Name resolution and discovery.* The discovery of Grid Services via an easy to use interface that enable rich queries to be submitted against metadata maintained in the registries, is important for the usability of the SB-DQP. The setup of SB-DQP requires the discovery of Grid Data Service Factories for importing the schemas of the participating data sources.

- *Service domains.* SB-DQP can be seen as a good example of service domains. It coordinates and orchestrates multiple Grid Query Evaluator Services and other Web services in a particular context during its lifetime.

- *Messaging and events.* There may be several contexts where SB-DQP needs to be notified of events. If the schemas of the participating data sources change the DQP would want to know about those changes so that the queries can be validated against the new database schemas. Another context is progress monitoring. When the query execution is in progress, the SB-DQP needs to receive notification messages that indicate the state of the execution at each query evaluation node. It is also required to receive regular updates on the state and availability of the computational resources, so that the query evaluation can be re-scheduled if needed.

- *Transaction.* Currently distributed transactions are not supported in SB-DQP, but it would certainly benefit from transaction interfaces provided by the infrastructure in the future.

- *Service orchestration.* SB-DQP implements a service orchestration framework in two senses: both in terms of the way its internal architecture handles the construction and execution of distributed query plans and in terms of being able to query over data and analysis resources made available as services. The latter form of service orchestration can be seen as complementary to other infrastructures, such as workflow languages.

6.7 Security considerations

The nature of the security challenges facing a GDQS are likely to vary from setting to setting, but may be quite demanding. For example, a single query may run over services within different domains of control, and could benefit from allocating evaluators to run on nodes that are under different domains of control. There may also be privacy issues on the data being manipulated by a query – for example, a requester may be reluctant ever to allow data from a private source to leave their organization, but may want to join that data with data from a public source. Thus single-enterprise, multi-enterprise and all-comers scenarios are all possible.

6.8 Performance considerations

There are many aspects to the performance of a distributed query. As queries are declarative, their execution must be planned. Query planning needs access to comprehensive information on the costs of using the services of relevance to a query, and also requires information on the computational resources available for evaluating a query.

Different operations in a query plan may prefer different forms of transport. For example, many distributed query processors support pipelined parallelism, but some operations are blocking, and thus may be more suited to bulk data delivery. Which operators should be used to evaluate a portion of a query will depend on the capabilities and load of the computational resources available. Parallelism can often be exploited to improve the performance of query evaluation, but scheduling is clearly challenging in an open environment such as the Grid.

6.9 Use case situation analysis

As stated in Section 6.1 each phase in the use case has implication on the required services/functionalities from the underlying infrastructure. The following list is an attempt to identify those requirements for each phase and to what extent they are met by the current OGSA reference implementation.

Service discovery and instance creation. The primary requirement here is the ability to discover the GDQS Factory and GDS Factories for the data sources by submitting a query to the service registries. This requires the service registries to support both the ability to specify and publish potentially rich information on the services being registered, and the ability to query this rich information using a well-known (easy to use) query language.

The existing OGSA reference implementation does not sufficiently support the ability to query against the service descriptions. The idea of Service Groups proposed in the latest draft of the GS specification provides more complete support in this regard.

Setting up the GDQS instance. One important requirement here is that the Grid Data Services must provide the schema of the database they wrap in a well-defined way. In other words the GDQS must be able to query the GDS instances to obtain the schema of a particular data source. Service Data Elements are one obvious candidate to provide such information in a well-defined way. Currently, querying this information via SDEs is not supported. GDQS obtains the database schemas by a custom extension to OGSA-DAI framework. The requirement referred to here, however, is more directly relevant to OGSA-DAI project rather than OGSA.

Collecting computational resource metadata. The relevant OGSA service here is the Index Service which is not part of the core OGSI but is provided as a higher level service. Although the Index Service seems to offer a flexible approach to collecting grid resource metadata, there are some issues that remains unresolved. The SB-DQP requires several classes of metadata to be interrelated and provided in a coherent way. The classes of metadata required are:

- The capability of a grid node (a machine that offers its computational resources to the grid user community) in terms of the CPU power, available memory, available disk space etc.
- Dynamic (real-time) information on the communication load on network connection between a set of grid nodes.
- The characteristics of a grid node in terms of the services it hosts. For example the information as to whether a particular grid node hosts a Grid Data Service Factory or a Grid Query Evaluation Factory.

Currently there is no a coherent way of collecting and relating such classes of metadata.

Query (request) submission. The implication of a query request in regard with the use of infrastructure services is that the GDQS has to dynamically create instances of GQESs on an arbitrary number of grid nodes to execute the sub-queries. Currently it is only possible to create a grid service instance on a node if its factory is already deployed on that particular node. This constraints the query optimizer to consider only a limited set of grid nodes (only those where a GQES factor exists). It is desirable to have the ability to dynamically ship the factory code to a hosting environment and deploy it so that any grid node can be considered for scheduling GQES instances.

Query execution and result delivery. The primary requirement here is being able to bind to efficient transport mechanisms. Currently only XML over SOAP/HTTP is seamlessly supported. The Reliable File Transfer Service that provides access to Globus Grid FTP APIs does not seem to be seamlessly integrating with the service interfaces. What is needed is direct support for efficient data transfer at the inter-service interaction level.

6.10 References

1. M.N. Alpdemir, A. Mukherjee, N.W. Paton, P. Watson, A.A.A. Fernandes, J. Smith, T. Gounaris, Grid Distributed Query Service (GDQS) Design, OGSA-DAI Design Document , 2, December, 2002.
2. J. Smith, A. Gounaris, P. Watson, N.W. Paton, A.A.A. Fernandes, and R. Sakellariou. Distributed query processing on the grid. Proc. 3rd Int. Workshop on Grid Computing, J.Sterbenz, O.Takada, C.Tschudin, B.Plattner (eds.), Springer-Verlag, 279-290, 2002.

7 Grid Workflow

7.1 Summary

Workflow is drawing attention as a convenient way of making new services by connecting existing services (like shell-scripts of UNIX systems). A new Grid service can be created and used by registering a workflow definition to a workflow engine. The definition is interpreted by the workflow engine, and calls several other Grid services as is specified in the definition.

7.2 Customers

Workflow will be used by both users and providers of Grid services. The cases when workflow will be used are as follows:

1. Connection of simple services: Users (or service providers) make a new Grid service by connecting several simple services (whose execution time is relatively short). For example, by connecting a stock information service and a currency exchange rate information service, a foreign stock information service can be made.
2. Job workflow: Users (or service providers) combine several jobs, specifying their execution order, input, output, etc. Here, jobs include both scientific and commercial jobs. For a scientific job example, simulation service and visualization service is connected using workflow. (Of course there are many other examples like compound simulation, data grid, etc.) Scientific job workflow may require huge amount of data transfer between services. As for commercial jobs, an example would be summing up sales result at each branch shop in parallel, and then collecting them at the head office.
3. Description of business process: Service providers describe business processes by connecting several services. For example, a travel agency connects a flight ticket reservation service, a hotel reservation service, and a vehicle reservation service to make a new travel reservation service. This kind of workflow is well investigated in the area of Web Services. Business process may take a long time (ex. one month) to finish, and may need exception handling mechanism (ex. cancellation of reservation).
4. System administration: Service providers describe a service for system administration using workflow. For example, a system administration workflow obtains an application program from an application repository using a file transfer service and deploys it to a Grid service container.

Combination of above examples is also possible. For example, one can think of a workflow which obtains weather information from various place of a country (above example: 1), and executes weather simulation job using the information and visualizes the result (above example: 2).

In addition, everything is abstracted as Grid service in OGSA. Therefore, everything which is abstracted as Grid service can be dealt with workflow.

A workflow definition itself should be seen as a Grid Service. Thus, workflow should comply with the various rules which the Grid Service Specification requires. For example, a workflow definition should have FindServiceData operation in Grid Service Port Type, and may need to support Notification; and a workflow instance should be created by a Factory.

7.3 Scenarios

As described above, workflow is used in various cases. Here, I will describe "application deployment scenario" in which typical relationship between other services/functions is shown.

7.3.1 Application deployment scenario

In this scenario, we assume that a system administrator or a user of a Grid system wants to deploy (install) an application to a Grid container.

The process is executed by a *service orchestration engine*. In the service orchestration, firstly, an application program is obtained from an application repository which may be implemented as *shared storage*. The storage may be found using a *discovery service*. If the storage has a functionality of *data cache / replication*, the program code can be efficiently obtained.

When connecting to the storage, *authentication / authorization* should be performed in order to restrict the access to the program. For authentication and authorization, a *policy management service* may be needed to get security policy for deciding if providing the program is allowed or not.

After obtaining the program, it is deployed using a deployment service which may be a part of an *administration service*. Here, *authentication / authorization* should be performed again. It may be needed to reserve the resource (the Grid container) beforehand using a *reservation service*.

All these processes might need to be logged using a *logging service*, and the log information might be passed to an *accounting service* for accounting. Again, for logging and accounting, a *policy management service* may be needed to obtain policies for them.

7.4 Involved resources

Computational resources are required in order to interpret and execute workflow descriptions. For managing long-lived workflow, non-volatile memories like files or databases are needed.

7.5 Functional requirements for OGSA platform

In the scenario described above, following functionalities are required.

1. **Workflow**

   With this functionality, several services are connected to realize application deployment. This functionality is represented as "Flow" in [References: 1].

2. **Discovery**

   In the above scenario, service discovery functionality is needed to discover storage service which contains the application program to deploy. This functionality is represented as "Discovery and brokering" in [References: 1].

3. **Shared storage**

   In the above scenario, shared storage is used as an application repository. This functionality is represented as "Data sharing" in [References: 1].

4. **Authentication and authorization**

   Obtaining application programs and deploying them into a Grid system may require authentication / authorization. This functionality is described in "Multiple security infrastructures" and "perimeter security solutions" in [References: 1].

5. **Application deployment**

   This functionality is required to deploy an application to a Grid container. This functionality is included in "Administration" functionality in [References: 1].

6. **Advanced reservation**

This functionality may be required to execute the application on reserved resources. This functionality is described in "provisioning" functionality in [References: 1].

7. **Logging and accounting**

   Processes like obtaining / deploying application programs might be logged, and the information might be used for accounting. This functionality is represented as "metering and accounting" in [References: 1].

8. **Policy**

   Authentication, authorization, metering, and accounting may require policies.

7.6 OGSA platform services utilization

1. **Service orchestration service**

   This service corresponds to "workflow" functionality, and is used as "workflow engine".

2. **Name resolution and discovery service**

   This service corresponds to "discovery" functionality.

3. **Security service**

   This service corresponds to "authentication and authorization" functionalities. In some cases, security is not implemented as services but functions attached to each service. However, some of the security functions such as decision of authorization may be implemented as services.

4. **Data management service**

   This service corresponds to "shared storage" functionality.

5. **Administration service**

   This service includes "application deployment" functionality.

6. **Provisioning and resource management service**

   This service includes "advanced reservation" functionality.

7. **Metering and accounting**

   This service corresponds to "logging and accounting" functionality.

8. **Policy service**

   This service corresponds to "policy" functionality.

7.7 Security considerations

There may be a need to deny access to workflow definitions from non-registered users. To implement this, authentication and authorization should be performed when creating a workflow instance using a Factory, and when accessing a workflow instance.
In addition, services called from workflow may require authentication and authorization. To support this, delegation mechanism like GSI may be needed.

7.8 Performance considerations

If execution time of a service called from a workflow is long enough, performance of a workflow engine does not matter much. However, if it is short, performance of a workflow engine may be important.

In addition, if there is need to transfer large amount of data between services called from a workflow definition, it is not efficient for a workflow engine to receive and send the data. Therefore, it may be needed to allow description of direct data transfer between services [References: 7].

7.9 Use case situation analysis

Many important works have been done in the field of Web Services. For example, there are WSFL[References: 2] by IBM, XLANG[References: 3] by Microsoft, BPEL4WS[References: 4] derived from both of them, WSCI[References: 5] by SUN, WSCL[References: 6] by HP. In the Grid computing field, GSFL [References: 7] was proposed by ANL. In addition, WFMC (The Workflow Management Coalition) is working in this field for a long time. These significant works can be a basis of a workflow specification of OGSA.

7.10 References

1. Foster, I and Gannon, D. The Open Grid Services Architecture Platform, 2003. http://www-unix.gridforum.org/mail_archive/ogsa-wg/doc00016.doc
2. Web Service Flow Language (WSFL 1.0), May 2001, http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf
3. XLANG Web Services for Business Process Design, 2001, http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm
4. Business Process Execution Language for Web Services, Version 1.0, July 2002, http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/
5. Web Service Choreography Interface (WSCI) 1.0 Specification, 2002, http://wwws.sun.com/software/xml/developers/wsci/
6. Web Services Conversation Language (WSCL) 1.0, March 2002, http://www.w3.org/TR/wscl10/
7. GSFL: A Workflow Framework for Grid Services, July 2002, http://www-unix.globus.org/cog/projects/workflow/

8 Grid Resource Reseller

8.1 Summary

It is not always desirable for owners of Grid resources to interface with end users directly. Inserting a supply chain between the resource owners and end users will allow the resource owner to concentrate on their core competence (e.g. in maintaining large supercomputers) and avoid providing costly interaction and support to a large number of consumers, allowing them instead to deal with a few large customers (potentially only one) who resell the resources.

End users can purchase resources bundled into attractive packages by the reseller (aggregation); these resources might in fact come from several resource owners.

The resellers can make money from reselling aggregated computational resources without having to own any resources themselves, thereby minimizing their own risk. In general, the reseller maintains resource provision by sustaining their relationships with upstream providers. However, to protect the agreed service level with the end users, the reseller may occasionally find it necessary to switch provider, either temporarily or permanently. Instead of worrying about maintaining resources, the reseller can focus on providing good customer care as well as marketing resource bundles to their target market(s).

This use case is adapted from the "Computational Reseller" use case, which was written by Jon MacLaren and William Lee, and appears in the GESA Use Cases Document [1].

8.2 Customers

There are three key actors in the Grid Resource Reseller scenario all of whom are customers of grid services in some form or fashion. The first of these is the **"Resource Owner"** of which there may be several in this scenario (which is considered from the point of view of the reseller). The Resource Owner is imagined to own resources which are expensive and rare, e.g. a supercomputer, although this does not have to be the case. These owners want to sell resources on in bulk, dealing with only a few large customers, who are resellers. They are interested in ensuring that they sell all their resource. They are less concerned about the actual usage of the resource which is the concern of the resellers, who are their customers. There will, however, be service level agreements between the resource owner and the resellers.

Next, there is the central actor, the **"Resource Reseller"**. The reseller acts as both customer (of resource owners, or upstream providers), and provider (to end users or downstream providers). The reseller need not be interested only in resource utilization, as their primary concern will be making a profit, i.e. if they can get all their customers to buy pre-paid resource usage packages (like "free minutes" on mobile phones). They do not care if these are ever used. In fact, a certain amount of overselling might happen, i.e. if everyone used all their pre-paid resources at once, the reseller would be in trouble. But this is extremely unlikely. A reseller will have service level agreements both with the providers and consumers of the resources. The reseller will have many more consumers than providers (e.g. an order of magnitude more), providing a natural fan-out as the supply chain moves from the resource owner to the end users.

Finally, there are the **"End Users"**, who are customers of a Resource Reseller. They are the real consumers of the resources. They do not know who owns the resources they use, as they get all their resources and associated service and support from the reseller. They will be free to select a reseller who is suitable for them - maybe based on the packages the reseller offers, and the package cost.

Naturally, the resource owner, reseller(s) and end users will be part of different organizations, and may be geographically distributed.

In the scenario presented below, we only consider a single reseller between several resource owners, and many end users. However, in considering the requirements for this scenario, it is important to envisage the possibility of a chain of resellers (as is the case for internet providers today).

8.3 Scenarios

As this use case is extremely general, there are many possible scenarios. Further, these examples are all similar, differing only in the details. Therefore, only one example is provided.

8.3.1 Computational Chemistry Reseller

Consider the example of a reseller who has strong links with the chemical industry and the expertise to support a wide range of chemistry applications running on supercomputers. To establish their business, they offer supercomputer owners the chance to sell resource in bulk to them, on the understanding that they will resell the resource. The reseller agrees to respect the policies of the resource owners when reselling. One resource owner provides cycles which are only for use by academic users; another offers a two-tier price structure, where cycles that are sold on to non-academic users are priced at a higher tariff. Both resource owners specify that the provided cycles must not be sold on to another reseller. Therefore the reseller decides only to deal with end users in this case.

As well as sourcing supercomputer centers, the reseller wants to provide access to all the popular chemistry packages. In some cases, the reseller can lease the licenses from the resource owners, some of whom have installed a subset of the target software. However, the reseller also sources some of these packages directly from the manufacturer, and must arrange for the staging (or installation) of the software on the target machines.

Finally, the reseller engages in a publicity campaign to attract users to its services. They market monthly packages of resources which include pre-paid ("free") items such as CPU cycles, secure and backed-up disk storage, and software licenses. To make itself as attractive as possible, the reseller deliberately resells the resources at a loss for the first three months of operation as a "not to be repeated" offer (loss-leading).

To facilitate the execution of the user's work, the reseller provides a resource broker. Users submit their work to the broker, which matches the user's preference with the policies of the resource owners. Based on this matching, plus information about the state of the resources themselves, the user's job is dispatched.

Where the reseller's service level agreement with the end user is "broken", the user may be entitled to some compensation. This may be described as part of the service level agreement itself.

It is useful to summarize the potential advantages of this scenario from the perspective of each type of actor:

1. The **Resource Owner**. There are a number of reasons why a supercomputing centre might wish to sell its cycles to a reseller.
    a) If all cycles are sold this way, the resource owner never needs to deal directly with large number of customers; this is useful as it is costly to maintain high quality of customer care. This policy enables them to manage their resources in a small number of large transactions.

b) During a period of low local usage, a centre might want to make a one-off sale of a large amount of otherwise redundant cycles.

c) A centre with seasonal peaks and troughs in local user usage might want to sell an amount of cycles (varying per month) to match expectation, thus maintaining steady usage.

2. The **Resource Reseller**. The reseller bundles the resources available to it from the various upstream providers, including some licenses it can obtain from the software vendors at a reduced rate (as it deals mainly with academics and in large quantity). An example offer is that for a reasonable monthly fee, the chemist gets 200 “free” CPU-hours on a Cray T3E, plus thirty uses of Gaussian98 thrown in (exceed that, and he gets charged quite a lot, of course.) They also include some compensation deal when jobs are not delivered due to downtime (a kind of insurance). A Reseller who has insights in the market trend can predict future demand and source resource provision from upstream vendors in advance when the price is attractive.
3. The **End User**. The chemist wants to get resources from the reseller because getting bundled resources reduces transaction costs in dealing with all parties manually. Also, he would expect to have better customer care and risks are shared with the reseller if upstream vendors default. Finally, the academic might be able to get his bundle for less because he gets it from the same reseller he gets his electricity / mobile phone time from. It encourages companies with existing micro-transaction technology (such as telecom, utility, etc.) to participate as resellers.

8.4 Involved resources

The *Resource Owner* is selling resources to one or more *Resource Resellers* (see also the GESA-WG Computational Provider Scenario [1]).

Each *Resource Reseller* in the supply chain is buying resources from one or more *Resource Owners* and upstream *Resource Resellers*. The reseller may bundle these resources before selling them to *End Users* or to downstream *Resource Resellers***.**

The *End Users* buy (possibly) bundled resources from the *Resource Resellers*.

Ultimately, it is the resources bought from the providers that are being consumed by the end users. This could potentially be any Grid resource. These resources could be geographically distributed, and could belong to a number of resource owners.

8.5 Functional requirements for OGSA platform

The presented scenario has many requirements, however, here we have chosen to describe those functions specific to the activity of reselling - i.e. we ignore generic requirements for work scheduling and execution which will arise from other use-cases. Here are the headings and functions from Section 3.2 of the OGSA Platform document [References: 2], required for reselling.

Discovery and Brokering

In the scenario, each reseller operates a broker to dispatch the user’s work to the available resources. The most important requirement here is that the broker can perform some sort of matching between the users’ preferences, and the resource owners’ policies (perhaps something like the Condor ClassAd scheme [References: 3]). Using the evaluated list of possibilities, the broker then uses information like acceptable turnaround time and cost to select specific resources for the work.

A reseller must be able to discover resource owners (or downstream resellers), and end-users must be able to identify resellers. Service Level Agreements must be agreed between these pairs of entities. However, in our scenario, these are infrequent (even once only) activities, and will be achievable through existing mechanisms such as networking, advertising, etc.

Metering and Accounting

The model for accounting and charging in the scenario is quite sophisticated. The Resource Owner will sell large amounts of cycles to one or more resellers. The price for these cycles will be negotiated between the two parties; it is unlikely to be uniform for multiple resellers. Further, whether cycles are used or not are not really the concern of the resource owner; some partial refund for unused cycles may be arranged between the two parties. In the situation of overuse, the resource owner would want to limit the amount of cycles that the reseller could use. Whether the resource owner would refuse any overrun, or whether overrun would be charged for at a far-higher rate, would be down to policy.

For the reseller, they must do their utmost to sell sufficient packages of resources to cover their expenditure, running costs, along with some profit margin. It should be possible for users to sign up for some sort of monthly plan, on-line, without human intervention. The reseller will need to bill the end users on the basis of usage, which is covered by existing plans in OGSA Platform. It is worth noting again that if the reseller obtains most of their money through contracts for pre-paid resource use, that they can oversell their resources (like hotel and airplane overbooking) to maximize income. Like the resource owner their income need not depend on the actual usage of the resources.

In terms of charging different granularities of trading must be supported. This also implies the ability to use different payment options such as purchase order/invoicing, Credit Card, etc.

There are several different charging schemes mentioned above. However, all the models described should be possible within OGSA Platform. Similarly, it should be possible for the accounting systems to operate autonomously for the vast majority of circumstances (including under usage and over usage). While the systems being designed in the GESA Working Group [References: 4] have cases like these in mind, it is hard to see how this functionality can be covered by the charging systems proposed in the OGSA Platform document [References: 2] (see Section 5.9 in particular); these seem to focus mainly on tariff-based charging, based on “accounting schemas”, and do not contain the concept of reselling.

Monitoring

The Resource Owner must be able to track the usage by the clients of the various resellers to check for resources being overused.

Policy

End Users and Resource Owners will have potentially complicated policies, as may the resellers. A reseller must not be able to sell on a resource in a way that violates the Resource Owner’s policy - e.g. selling cycles to an industrial user at an academic rate. Similarly, a reseller should not be able to run a user’s work on resources which violate their policy, e.g. running a job from a user with an “environmentally friendly only” policy on a computer owned by a corporation frequently responsible for pollution, etc.

There must be some way in which to aggregate the policies of all upstream providers.

Extended Service Level Agreements

This is not a heading in OGSA Platform, but it’s something that is needed in this scenario and other GESA-WG use cases [References: 1]. We want to incorporate cost information into the SLAs between parties. In certain circumstances, we would also like it to be possible to define rates of compensation in the SLA – For e.g. if the user can’t access their pre-paid resources for 24 hours or more in a month, they will be refunded £2, etc. This is the subject of ongoing work within the GESA-WG group [References: 4].

8.6 OGSA platform services utilization

The following services (or interfaces if appropriate) are necessary to provide functions in the previous section.

1. **Policy** [10]

   The scenario described here has sophisticated requirements for policy definition and handling within OGSA Platform. In particular, we have a need to aggregate several policies within a supply chain.

2. **Metering and accounting**

   This interface will need to be made more flexible if it is to cope with the requirements of the scenario described in this document.

3. **Provisioning and resource management**

   Required for SLA agreement and monitoring. This functionality will need to be able to handle the extended SLAs discussed in the previous section.

4. **Brokering** [11]

   Brokering functionality is required. The policy matching aspects of this are probably to be handled by the Policy interface.

5. **Monitoring service** [12]

   This service is used for monitor function.

8.7 Security considerations

The Resource Owner and Reseller chain should be able to provide the user with assurances on privacy, where this is required.

8.8 Performance considerations

Where the reseller chain is a few steps long, it should still be possible for the user to get good performance when accessing the resources.

8.9 Use case situation analysis

We do not believe that there are any examples of this use case in the Grid. (Although Application Service Providers exist, these also own the computational resources used to process the work, and so do not qualify as Resellers.) Of course, there are hundreds of examples in other areas, most notably internet provision and mobile phone provision. We are confident that once the enabling technology is present, that reseller businesses will be established.

8.10 References

1. Keahey, K., MacLaren, J. and Newhouse, S. (Eds.), “GESA Use Cases”, February 2003. http://www.doc.ic.ac.uk/~sjn5/GGF/draft-ggf-gesa-use-cases-01-7.pdf

[10] The explanation of the policy interface in [References: 2] is very vague and is not clear what it is.

[11] This function should be added to the OGSA platform interfaces.

[12] This function should be added to OGSA platform service.

2. Foster, I. and Gannon, D. (Eds.), “The Open Grid Services Architecture Platform”, February 2003. http://www-unix.gridforum.org/mail_archive/ogsa-wg/doc00016.doc
3. Raman, R., Livny, M. and Solomon, M. “Matchmaking: Distributed Resource Management for High Throughput Computing”, *Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing*, July 28-31, 1998, Chicago, IL.
4. Grid Economic Services Architecture Working Group (GESA-WG). Home page: http://www.ggf.org/3_SRM/gesa.htm

9 Inter grid

9.1 Summary

This use case is similar to commercial data center in many respects (it needs all the features of commercial data center). But additionally this use case looks at grid running at its full potential in a geographically distributed company with access to the outside world (not just in a specified data center). To do that two verticals are selected aerospace and financial industry. But the use case is applicable to any other vertical like telecom, manufacturing and so on. It emphasizes the following features - plethora of applications that are not really grid enabled and is difficult to change, mixed grid and non grid data centers, grid across multiple companies that agree to collaborate, interoperability with web services standards, a combination of compute/service/data working as a single whole. It brings in generic concepts of the utility computing into view for grid to successfully operate in industrial environments. It also takes into account the industrial concerns like migration, mixed operating environments, maintenance requirements etc.

9.2 Customers

Grids in industry verticals (in manufacturing, telecom etc). For e.g. aerospace, automobile

On a user basis the main roles are Grid administrator, IT system integrator, Business activity Manager (There is also the every day Grid user, but his requirements are included in the business activity managers requirements). These roles are defined in the commercial center use case.

9.3 Scenarios

We take the examples of aerospace and financial industries here to illustrate the use case.

The operating environment in financial industry comprises of data centers (Linux, UNIX flavors, mainframe systems, storage mgmt systems across geographies) and user machines (PCs, workstations). Grid has to interface with windows XP based servers and Mainframes in its entire feature set as it does with UNIX flavors and Linux.

Migration is a very important parameter for financial industry. Many financial applications and systems are legacy – that means they are not grid enabled and don't follow any specific models. The industry may not consider grid as an option for data center unless a safe migration path is defined that enables minimum downtime of applications and systems. Additionally non grid applications has to be accessed by grid applications as in a normal grid and the performance of non grid applications cannot be degraded (can be enhanced).

Typically a large aerospace company has many data centers and geographical sites. The industry also has a large number of scientific and high performance computing industrial applications that reside and exist alongside business applications. Some of the business applications and even some of the computing applications are legacy and very difficult to change. These maybe installed in grid enabled systems as well, although not designed to derive any particular benefit from it. Hence a mixed environment of grid and non grid networks and systems will be employed. This has additional complications in geographical and virtual separation of systems, user access management and so on. This brings in the generic utility computing requirements for the data centers where the applications are not aware of the grid infrastructure but still are able to get flexed resources if needed. The grid administrator on the other hand has perfect freedom to copopulate his system with nongrid environments as well.

A large number of applications (especially on the business side) will be using and accessing web services. Further there might a full fledged web services environment coexisting in the grid environment. *Therefore there is a need for coexistence between grid and web services and conflicting requirements or environments are a set back.*

Typically in a large aerospace company there are design centers that may be outside the firewalls of the company, but with whom there are contractual agreements for collaborative design. Therefore there is a whole issue of control of systems and applications by the correct entity, identity management and in general issues associated with all levels of security in a fully collaborative environment by two companies that want to protect their IP and resources at various levels of access. Grid security and collaboration mechanism should allow this. This again brings the utility computing concept into the view since all kinds of situations can occur from a security perspective. So it becomes imperative to control and flex the layer 2 devices, firewalls and other non computing devices in the IT network to concur with the strict security requirements.

All kinds of grid environments are possible in an aerospace company. This includes compute grids, data grids, and service grids and any other nomenclature based grid identification schemes. These need to coexist. Some examples for these are given below to illustrate that there will be grids with different purposes

Compute grids:

Very intensive computing is involved. For e.g.: Computational Fluid Dynamics (CFD): aerodynamics, “Structures”: stress testing, crash simulation, bird-strike simulation.

Some of these are bounded problems which can be parallelized and run adequately on distributed memory message-passing architectures. These are “utility computing feasible”- that means capacity on demand feature is equally important with the other capabilities provided by grids like virtualization or resource sharing (using existing PCs or workstations that are not part of a data center).

Data grid:

For e.g.: simulation of wing structure analysis could require a 1 Terabyte memory single system image. This could potentially run on 64xcpu Superdomes (high end Hewlett-Packard systems given as example) and creates 20 terabytes of data (highly compressed) on stresses and strains throughout the wing structure. So this level of data handling capability is very important from a Data grid perspective. These major simulations need changes to the OS or middleware that can support these kinds of operations specifically to address the 1 Terabyte memory requirement[13] and huge amount of data redundancy and synchronizations needed. That in turn translates to grid middleware and API requirements for customizations that individual customers need to carry out.

Service grid:

A grand challenge is want to be able to simulate the whole aircraft systems which are otherwise decade long major projects. financially it makes sense to do away with as much physical testing as possible with numerous outsourced and collaborating design consulting companies (For example - 12 risk-sharing partners for an existing project). Therefore different computing activities (for example tests/interaction to one consulting company) are drawn up as services with service agreements. Through interplay of web services and grid the computing service is executed over multiple sites/companies. Additionally buying and selling extra compute cycles from/to a third party utility provider is also considered (Grid resource reseller use case). Difficult problems that need to be solved are

- Security guarantees between different companies that get mixed up,

- Management of resources to fulfill all the different service requirements detailed above (this includes configuration, figuring out the quantity of resources needed)
- Guaranteeing mechanisms to satisfy reliability and meeting all the different levels of SLM requirements (at a resource level, to a service level)

Interoperability between grids is a key requirement. Therefore some sort of interoperability standards need to be there. Different data centers in one company might be using different grid middleware.

Maintenance is another strict business requirement – things like usability and troubleshooting mechanisms are relevant in this regard. *Upgradation of grid software without bringing down all systems is a necessity*. People interfacing to grid has to usable – might enforce some guidelines on user interfaces to grid.

9.4 Involved resources

The resource needs assume the commercial data center use case resource needs as a base. Additional emphasis is placed on the needs of handling non computing resources in the grid: For example devices with a software interface (aircraft systems - the grid concept is applied and this resource is shared, especially for testing and modeling by different teams.), firewalls, router configurations etc.

These resources are geographically distributed inside a company. There is multiple user access to these resources (with varying levels of access).

9.5 Functional requirements for OGSA platform

Commercial data center use case is the base. Additional requirements are listed below.

- *Discovery and brokering*. Ability to discover and broker services that are across organizations with various levels of security being considered
- *Metering and accounting*: Access to heterogeneous storage systems; preferences for interface format to such systems ; Accounting requirements, including information on the requirements for dealing with multiple accounts and/or accounting systems
- *Monitoring*. A global, cross-organizational view of resources and assets with emphasis on life cycle management and fault handling. Automated actions are necessary, and hence the data should be sufficient for that. There needs to be APIs to let this information elsewhere as it's residing in a mixed grid/nongrid environment.
- *Provisioning*. Compatibility to nongrid provisioning systems due to the mixed environments. A way out is specifying APIs to interact with Grid provisioning APIs.
- *Resource collision resolution*. In a fully mixed environment there is possibility of resource usage and management of collision of same resources or concepts (like VO). This needs to be resolved (at least guidelines ought to be there).
- Usage models that provide for both batch and interactive access to resources.
- Support for the management and monitoring of resource usage and the detection of SLA violations by all relevant parties.
- *Load Balancing:* An additional requirement is dynamic consideration of security requirements along with this.

- *Legacy application management*. Legacy applications are those that cannot be changed, but are too valuable to give up or to complex to rewrite. Grid infrastructure has to be built around them so that they can be used as well by the nongrid user.
- *Administration*. Be able to "codify" and "automate" the normal practices used to administer the environment. *The goal is that system should be able to self-organize and self-describe to manage low level configuration details based on higher-level configurations and management policies specified by administrators.* There is also the all important *migration issue* for services that are not grid enabled to grid enabled services. Some usability guidelines might be required to ensure easy usability of grids. Grid software upgradation should not enforce a complete grid environment downtime but should be done partially system by system or without downtime at all.
- *Programming model*: guidelines and methods to grid enable applications (including legacy applications – could end up being difficult for the first revision. For e.g.: when applied to programs that are multithreaded or multiprocessed).
- *Program execution*: specified tolerance of application and network delays. Scheduling of priority to get application real time requirements (resources, messages etc). Standard job description (environment, job etc)
- *Logging*: The logged information should be enough for detailed troubleshooting. This might be mean varying levels of logging as required by administrator.
- Policy (from a security and identity point of view)
- Collaboration requirements (various levels of security to control the access of a resource or a service)
- A number of user interfaces will be required. Such interfaces will be required to provide the user with the ability to submit, monitor, and steer runs. In addition, it would be helpful to have an interface which provides information for administer and for performance-tuning, allowing to audit the computation, compare different runs in terms of resource usage, and provide information about each run including what version was installed on the computing resources, how different resources performed, accounting information, etc

9.6 OGSA platform services utilization

Since this use case covers a vast area of application of grids, all services in the OGSA architecture specifications is needed in this case.

9.7 Security considerations

Commercial data center use case requirements are the base to start with. Additional to that is the use, access and flexing of non computing resources like software interfaced systems, routers, firewalls and so on. To successfully operate in a fully fledge inter grid environment these will have to manipulated and allocated as desired. Security then becomes the prime factor for the administrator to consider here.

9.8 Performance considerations

Again the commercial data center requirements are the base here. The caveat being that speed of execution is also a parameter considered in some parts of the Intergrid where the scientific applications are running.

From a gross performance requirement the CIO wants to be able to quickly deploy the corporation's resources to the critical problems at hand. Today for example in a financial industry it's a manual process which can be 13 weeks or more in some examples. As the company consolidates data centers, the potential loss of a data center looms large. The ability to redeploy critical customer facing services to another facility is also critical. So migration and deployment has to be with industrial quality mission criticality.

To adequately manage the performance of the system, monitoring and forecasting requirements prior to and during application execution are required for identified systems. Some of these are

Network bandwidth, Latency and jitter, CPU load, Information service query time, Disk capacity, speed, Multicast performance, Remote memory and data sizes and access times, application timings, CPU speeds (specs and benchmarks)

Some of these can be collected using existing performance management tools and some of these (the ones required for a dynamic and automatic tuning) will have to be supplied by grid system or API based tools.

To get a feel for the real world performance and fault management requirements in the intergrid scenario consider a financial company where about 40000 servers could be used along with the 120000 agent office machines - this while spread across 19 data centers, shrinking to about 4 over the next few years. They are driven by their ability to manage the complexity of deploying an ever growing set of services without additional people (migration and performance requirements). The disaster recovery expectation is to be able to lose a data center and have the critical service up in less than 24 hours (Currently 7-10 days). Reliability requirements are a downtime that can be measured in minutes in a year.

9.9 Use case situation analysis

This is clearly in the research phase. Industrial and business applications running in a grid environment is an uncharted territory. Once the commercial data center becomes a success a proper company wide industrial strength grid (inter grid) can be attempted.

9.10 References

1. Forrester 'Organic IT' report

10 Interactive Grids

10.1 Summary

In addition to batch processing, interactive processing can be envisioned as a useful application of Grid technology. Applications that today are restricted to running on a single platform might be made to run on a network/grid by using the concept of an interactive grid and thus allow individual user interactions. That means that the applications and spawning processes should know where to send the job parameters and execution results (or a middleware agent should route them), and the user interface should be able to interact and synchronize seamlessly with processes that have been farmed out or restarted. Compared to the online media use case, this use case emphasizes a very high granularity of distributed execution (thread-based, or even procedure-based, depending on the customer scenarios).

Another aspect of interactive grid is the ability to schedule and perform work, based on an automated schedule, and to do automated series of actions. This also means that a job is being controlled dynamically by an external agent which may or may not be a human.

Computational steering[14] is another aspect of interactive grids that follows from this. This means that the grid user has the capability to steer his/her computations and resource needs interactively and dynamically during runtime. This also means that user can access new grid locations and ask to change underlying physical resources dynamically during normal interactive proceedings.

10.2 Customers

This use case is applicable to the following types of customers:

- Individual grid users who are oblivious of whether their application is running in a grid or a non-grid environment. This is especially important for virtualization of future applications and usage scenarios, as application vendors don't have to design specifically for being part of a grid environment, for the applications to become virtualized.
- Small and medium business customers who have limited computing resources and have applications that do not require huge resources, but do need more than single PCs.
- Grid users who have graphical interfaces (as complexity increases all user interfaces tend to be graphical and hence require synchronization and enable interactive use).
- Users of legacy applications that today run on a specific platform, but in the future will be able to run on the network (grid) without the code being grid-enabled.

10.3 Scenarios

1. UI based operations controlled by a grid user.
2. Pure parallelism (to any granularity) and pervasive computing[15] (not just batch jobs).

Traditionally, scientific and academic users have submitted work to Grid computing systems in batch jobs. For their purposes, the time delay while awaiting results from batch jobs has been acceptable. However, as Grid computing becomes more of a tool for commercial markets, users are expected to want to be able to monitor and manipulate results in real time.

[14] Computational Steering can be defined to include modifying program states, managing data output, starting and stalling program execution, altering resource allocations, changing underlying resources etc. Dynamic steering requires the user to monitor program, environment requirement, system state and have the ability to make changes.

In the current batch model, each computing job is submitted to a Grid management system with instructions for the task and requirements for computing resources. The Grid management system allocates resources, completes the job and sends back the results—the user cannot review intermediate results, and cannot submit changes until the next batch job submission. With the new interactive grid model, users could have results delivered in real time via graphical displays, allowing for adjustments, manipulations and data changes to the job while it is still in process. There is a computational steering aspect in this.

Interactive technology for Grid systems is useful for reducing run time and improving results for a broad range of compute-intensive applications, including graphics visualization and rendering, engineering applications such as CAD/MCAD, digital content creation, streaming media, video games, text editing and e-mail applications. In addition, the remote access enabled by interactive Grid technology can deliver cost savings by limiting the number of licenses necessary for expensive, specialized software.

In terms of parallelization, process-level, thread-level and even instruction-level parallelization can be visualized in grid by introducing the interactive grid concept.

10.4 Involved resources

This use case only requires the use of generic computing resources (CPU resources and various operating systems running on them, and storage resources).

10.5 Functional requirements for OGSA platform

The commercial data center use case is assumed to be the base here to avoid rewriting every functionality. However the dynamic nature of interactive grids brings in new requirements to traditional services from the perspective of fine grained transience and virtualization of interaction. Some of these capabilities required are.

- *Discovery and Brokering.* These functionalities should be able to recognize those grid resources that can support interactive grid functions. It must be possible to make real-time adjustments to resources based on job requirements and user input
- *Metering and accounting*. It must be possible to measure and account for resource usage. This is more complex in a computationally-steered model of execution than in the preordained model of a batch environment.
- *Monitoring*. Monitoring Agents - stand-alone software agents launched by the interactive Grid middleware to monitor security and performance, so that, for example, SLAs can be enforced.
- *Data sharing*. Data archives and caching data capable and managed for consistency per user per job interaction.
- *Policy*. It is important to be able to represent policy at multiple stages in hierarchical systems with a view to automating the enforcement of policies that might otherwise be implemented as organizational processes or managed manually.
- *Transport management*. It can be important to be able to schedule or provision bandwidth dynamically for data transfers, or in support of the other data sharing applications.
- *Session Management*. For maintaining job performance, including enforcement of SLAs and QoS requirements. Hierarchical sessions are supported - sessions that can have global scope, individual layer scope, and sub sessions inside that to track unit computations described by policies.

10.6 OGSA platform services utilization

This use case is currently being implemented, and hence we do not yet have a full understanding of all the services that will be required. However, an initial list based on current experience is given below.

- *Name resolution and discovery service*: Capability to differentiate devices that can participate in an interactive grid.
- *Data management service:* Has to take into account the data caching, and managing consistency per job per user. Additionally, it must be possible to schedule bandwidth dynamically for data transfers.
- *Fault handling service:* Fault handling per job per user.
- *Policy service**:*** Policies and policy handling at multiple stages in hierarchical systems.

10.7 Security considerations

Security mechanisms must allow for session-based action series. There is a hierarchical session model, and the security for these has to be coordinated to prevent unauthorized access and malicious use. So admission control for global and individual sessions should be possible.

10.8 Performance considerations

Matching of resources to user requirements on a much more dynamic scale than for a batch grid will introduce performance issues.

10.9 Use case situation analysis

This use case is currently in the research phase. Hewlett-Packard has produced the initial versions/demos of this.

10.10 References

11 Grid Lite

11.1 Summary

This use case extends the use of grids to small devices – PDAs, cell phones, firewalls etc. The key requirement is to identify a set of essential grid services for PDAs, for example, that enable the device to be part of a grid environment. Grid software components running in a "grid lite" environment need to have a smaller footprint, and generally to be more efficient, than would be necessary for a "normal" computational grid node. With a grid lite infrastructure in place, other grid applications and users would be able to run tasks on these grid-enabled PDAs or other smaller footprint devices, and vice versa.

Layer2 or 3 devices that have more of a firmware interface (embedded operating systems) should also be able to be virtualized and to be grid devices. The main requirements here are virtualization and pure software-based remote configuration/provisioning.

Mobility has many issues with connectivity/virtualization and synchronization/interactivity.

11.2 Customers

This use case is applicable to the following customers:

- Individual grid users who use PDAs.
- Companies that manufacture small devices or network devices or other non-computing devices like printers.
- Small and medium business customers who have limited financial resources, who would like to virtualize their environments completely but cannot do so due to the presence of layer2 devices like firewalls or routers in the network.

11.3 Scenarios

To be supplied.

11.4 Involved resources

This use case involves PDAs, cell phones, appliances, Layer2 devices like firewall or other network devices.

11.5 Functional requirements for OGSA platform

The requirements that are specifically applicable to this use case, and may not be covered in other use cases, are:

- *Discovery and Brokering.* Discovery mechanisms and registry mechanisms for layer 2 devices and transient devices. Ability to handle a very large number of resources.
- *Monitoring*. Monitoring model that incorporates a synchronous and asynchronous model (for offline processing and mobile processing).
- *Data sharing*. Data archives and caching data capable and managed for consistency for offline and online actions.
- *Proxy grid client mechanism*: For many devices it may not be acceptable to have a grid client running in them. In these cases a proxy mechanism would be needed.
- *Small footprint essential service group*: The grid services groupings should take into account small-footprint devices.

11.6 OGSA platform services utilization

The preliminary service requirements for grid lite are listed below. More details should be forthcoming as grid-lite infrastructures are implemented.

- *Name resolution and discovery service*: Efficient naming and discovery of layer2 devices, PDAs, and transient devices such as mobile phones. In the case of mobile devices the issues of discovery associated with moving away from local environments is also important.
- *Security service*: Like discovery, security has to tackle the issues related to devices that can have no security, and with devices that can have mobility, as well as VPN-based connections. Reconfiguring firewalls is another issue here.
- *Scheduling service*: Handling transience.
- *Brokering service***:** Handling transient devices.
- *Data management service:* This service has to take into account the often very low bandwidth available for communication (coding schemas and so forth).
- *Provisioning and resource management service*: Provisioning layer2 and devices such as PDAs.
- *Fault handling service:* Fault handling of grid lite devices.
- *Policy service***:** Policies and policy handling of grid lite devices—especially policies for mobility and offline synchronized actions.
- *Monitoring service:* Monitoring of grid lite devices (monitoring principles may be different for transient devices and layer2 devices).

11.7 Security considerations

A VPN-based security model should be acceptable. Devices with null security mechanisms will have to be able to work as part of the grid if desired.

11.8 Performance considerations

There may be bandwidth issues if large amounts of data must be transferred to main grid activity.

11.9 Use case situation analysis

This use case is currently in the research phase.

11.10 References

To be supplied.

12 Virtual Organization Grid Portal

12.1 Summary

Given that the grid enables people to be members of many VO's and each VO gives one access to various computational, instrument-based data and other types of resources, it is very natural for these VOs to produce a Grid portal which provides an end-user view of the collected resources available to the members of the VO. By producing a portal with "one-stop" shopping for users who participate in a VO, the VO makes its resource much more useful and accessible for their users.

These grid portals have several elements in common:

- Provide a public face for the VO with various outreach and informational materials
- Provide a set of collaborative tools (discussion, file storage, calendar, announcements, etc.)
- Provide access to any large data stores which are available to the members of the VO
- Provide the ability to make use of any computational resources available to the members of the VO

These portals are usually a combination of web-based and other tools. Typically, essential functionality is provided via grid-enabled web servers while more sophisticated tools are deployed to users' desktops.

Given that there are a number of common elements which can be reused across multiple grid portals, and to simplify the user's experience as he/she moves from one portal to another, it is important to develop best practices and techniques for the development and deployment of Virtual Organization Grid Portals.

12.2 Customers

The customers of this capability are effectively any virtual organization which intended to provide a user-facing component to their resources. In many ways, the Virtual Organization Grid Portal is a capability which can be used by many of the other scenarios described in this document. This scenario does not describe the particular portals for the other scenarios, but instead focuses on the common tools and capabilities which may have uses for any Virtual Organization Grid Portal.

12.3 Scenarios

There are an increasing number of grids where the focus is collaboration centered on some scarce physical resource. Often these resources are so large or so expensive that there can only be a very small number of installations across the world. Some of the examples of this type of collaborative activity include Astronomy, High Energy and Nuclear Physics, fusion research, earthquake engineering and others.

These broad collaborative efforts generally have the following attributes:

- Geographically dispersed access to computation, data and instruments
- The need for environments for participants to meet and work together across large geographical distances

Most of these collaborative activities are by their nature world-wide and cross-organizational. Within the collaboration there are many groups of varying sizes which are dynamically formed to work on a wide range of problems including experiment design, experiment scheduling, equipment operations, management, publication of results, and many others. All of these groups must operate with members scattered around the world in any time zone.

For these collaborations it is very important to maintain the security of the data, ideas, and the interactions of each group. While there is overall collaboration in the use of the equipment, there is often competition between subgroups within the collaborations in their pursuit of research results. In addition proper security and access control are absolutely necessary when dealing with the control and operation of any type of experimental equipment or the monitoring of the real-time data as it comes from the experimental equipment.

12.4 Involved resources

The Grid Portal can provide interfaces and access to any type of Grid-enabled resource which is within the purview of the Virtual Organization. These can range from computer resources to physical sensors and data resource. They can be centrally located or widely distributed.

12.5 Functional requirements for OGSA platform

Because of the cross-cutting nature of this scenario, the functional requirements on the OGSA platform cut across all of the services described in OGSA platform document.

12.6 OGSA platform services utilization

Because of the cross-cutting nature of this scenario, Grid Portals have the potential to utilize all of the services described in OGSA platform document. Virtual Organization Grid Portals will place particular strain on the security capabilities of the OGSA platform, as described in the next section.

12.7 Security considerations

As membership in multiple Virtual Organizations becomes a desirable and feasible situation, and as increasingly broad user populations interact with the grid, there are a number of new issues which will come to the forefront and need solutions. These fall into two broad categories:

- Security proxy capabilities
- Credential management issues

Security proxy capabilities are a significant but somewhat short-term problem. To understand the need for security proxy capabilities, imagine that a grid portal would like to allow its users to use WAP on a cellular phone to monitor a batch job and possibly steer the batch job in some way. For the foreseeable future, it is not likely that the cellular phone will have complete support for OGSA protocols and services. To allow the cell-phone user to perform operations within the Grid there will be a need for a proxy which talks the WAP protocol to the cellular phone and the Grid protocols to the rest of the Grid.

Some day in the future, this will not be necessary when all devices support OGSA services and protocols in a native way.

Credential management is related to security proxy, but different in some important ways. Much as the cellular phone is not capable of running the Grid protocols directly, it is also not capable of carrying Grid credentials around to properly establish identity. As such, an intermediate mechanism is needed which is capable of handling the user's credentials.

The problem is further complicated as users join perhaps thousands of virtual organizations, each possibly with different credential mechanisms and credential authorities. At some point, the management of these credentials becomes completely unwieldy. This is especially the case if a person is a mobile user migrating between different workstations throughout his/her day. It is not practical to install several hundred credentials in every piece of equipment that the user may use throughout the day before they can use the equipment. Beyond the inconvenience of installing key material, as one moves around, there is the grave danger of leaving key material in a place where it may be compromised.

The ideal solution for this is to use "smart cards" which can contain key material in such a way that it is not actually placed on the computer which the user intends to use. The unfortunate situation is that smart cards are effectively not supported at all by any commodity hardware/operating system combinations.

The net result is that we will need a mechanism for the management of user credentials. The MyProxy [reference] mechanism which is currently available is a basic mechanism, but requires moderate user sophistication to manage and use their credentials. In addition, users still must keep track of the location and purposes of each of their credentials.

These security problems are not unique to the Virtual Organization Grid Portal, but as organizations are increasingly able to quickly and easily deploy portals, these problems will quickly become very important.

12.8 Performance considerations

Grid Portals generally do not have significant issues in terms of performance. However, there is often a situation where a Grid portal must act as a proxy between a non-grid-enabled tool and a resource which is available using Grid protocols. Some of this proxy activity is short-lived and is transactional in nature. Other proxy activity may need to be maintained for a long period of time such a running a Brew [reference] application running in a cellular phone which needs a proxy to a subscribed OGSA service.

12.9 Use case situation analysis

The primary unmet needs of the Virtual Organization Grid Portal fall into two basic categories:

- The need for enhanced security and credential capabilities as described above
- The need for high level services which reflect a "user view" of underlying services

To understand the need for "user-centric" services, we can look at the GridFTP capability in the Globus Toolkit and compare the GridFTP API used by programmers with the command line program globus-url-copy. The GridFTP API is very powerful and flexible and exposes all of the capabilities of GridFTP to a sophisticated programmer. The globus-url-copy command (at its simplest) takes two parameters in the form of URLs and copies data.

As we move towards Virtual Organization Grid Portals, we will increasingly need access to OGSA services which provide simple, high-level functions more akin to the globus-url-copy command than the GridFTP API. Virtual Organization programmers will need to write small applications which are capable of easily composing several of these high level services to accomplish some new task. These applications may be written in languages such as JSP, Perl, TCL/TK, etc., rather than JAVA or C. It is entirely possible and desirable that the writers of the low-level (powerful/flexible) services will also provide these high-level services. The advantage of both services being implemented by the same group is that the higher level service is a natural mechanism to test the lower level services.

There are a number of existing efforts which can be viewed as early analogues for this concept. The Globus Toolkit COG [reference] is an example of encapsulating Grid functionality in an "easier-to-use" form.   The COG enabled the creation of simple Grid tools in a variety of simple languages. Another early example of this type of effort is the Grid Portal Development Kit (GPDK) which encapsulated high level grid functionality in a set of JAVA beans which enabled development in the JSP language.

It is important to note that the key need here is not the particular implementations in these languages/environments, but instead the services which provide the high-level user-oriented functionality which will allow a wide range of portal toolkits to be developed using those services. These services can be thought of as a layer which is built on the more fundamental OGSA services.

12.10 References

[1] A Toroidal LHC Apparatus (ATLAS) http://atlas.web.cern.ch/Atlas/

[2] Brew, http://www.qualcomm.com/brew/

[3] Commodity on the Grid (COG), http://www.globus.org/cog/

[4] Compact Muon Solenoid (CMS) http://cmsinfo.cern.ch/Welcome.html

[5] DZero http://www-d0.fnal.gov/

[6] European Virtual Observatory (EVO)

[7] The George E. Brown Jr. Network for Earthquake Engineering Simulation (NEES) www.neesgrid.org

[8] Japanese Virtual Observatory (JVO) http://jvo.nao.ac.jp/

[9] Laser Interferometer Gravitational Wave Observatory LIGO http://www.ligo.caltech.edu/

[10] MyProxy, http://grid.ncsa.uiuc.edu/myproxy/

[11] National Fusion Collaboratory (FusionGrid) http://www.fusiongrid.org/

[12] Sloan Digital Sky Survey (SDSS) http://www.sdss.org/

[13] US National Virtual Observatory (NVO) http://www.us-vo.org/

13 Persistent Archive

13.1 Summary

We build many large-data scientific preservation environments using the capabilities provided by virtual data grid technology (e.g. California Digital Library, NARA persistent archive, NFS National Science Digital Library). Preservation environments handle technology evolution by providing appropriate abstraction layers to manage mappings between old and new protocols, old and new software systems, and old and new hardware systems, while maintaining authentic records. Preservation environments typically organize digital entities into collections. Authenticity is tracked by the addition of appropriate metadata attributes to the collection to describe provenance, track operations performed upon the data, manage audit trails, and manage access controls. Validation mechanisms are provided to check that the data has not changed.

Virtual data grids provide two necessary capabilities:

- Support for the creation of a "derived data product" from a specification. Derived products can be a "transformative migration" of a digital entity to a new encoding format, or even the application of the archival processes that are used to create an "archival form" of a collection.
- Management of the completion state associated with the execution of a service. Note that the "completion state" that describes the result of the application of "archival processes" must be preserved in order to check authenticity.

Persistent archives differ from virtual data grids in that in addition to an "execution state" that is transient; a "completion state" is preserved. Persistent archives build upon standard remote data access transparencies:

- logical name space to provide location independent naming convention
- Storage repository abstraction to characterize the set of operations that are performed on remote storage systems (file systems, archives, databases, web sites, etc.)
- information repository abstraction, to characterize the set of operations used to manage a collection within a database
- Access abstraction, to characterize the set of services that are supported by the persistent archive.

Preservation environments support archival processes, used to create the archival form of collections. The archival processes include:

- Appraisal – analysis of which digital entities to preserve
- Accession – the managed ingestion of digital entities into the data grid. This corresponds typically to a registration step, and then a data transport step
- Arrangement – the creation of a hierarchical collection for holding the digital entities
- Description – the assignment of provenance and authenticity metadata to each digital entity
- Preservation – the creation of archival forms through transformative migrations, and the storage of the data
- Access – support for discovery and retrieval of the registered digital entities.

13.2 Customers

Equivalent technology is needed by all groups that assemble large data collections, or that try to manage a collection for a time period greater than 3 years (the time scale on which technology becomes obsolete). Users include NARA, Library of Congress, NHPRC state persistent archives, NSF NSDL, NVO, NIH BIRN, NASA ADG, NASA IDG, DOE PPDG, etc.

When dealing with scientific data, three capabilities are needed in particular:

- Support for parallel I/O, to send data effectively without having to optimize the window size and the system buffer size
- Support for bulk operations, including registration, loading, unloading, deleting.
- Support for remote proxies, for data subsetting directly at the remote storage repository, for metadata extraction, for bulk operations

Every community we work with is dealing with small data sets (size less than the network latency * Bandwidth delay product). In aggregate, their data is measured in tens of terabytes to petabytes. An example is the 2 Micron All Sky Survey, a collection of 5 million images totaling 10 TBs of data. The images are registered into a collection, aggregated into containers, and stored into the HPSS archive. Containers were used to minimize the number of files that were seen by the archive. At SDSC, the archive contains over 700 TBs of data, but only 17 million files. The addition of 5 million names to the HPSS name space for only 10 Terabytes of data was viewed as unacceptable. By aggregating the images into containers, we stored the 10 Terabytes in 147,000 “files”. Since we sorted the images when they were written into the containers, such that all images for the same region of the sky were in the same container, it then became very easy to support the construction of mosaics.

An example of the use of remote proxies is the Digital Palomar Observatory Sky Survey. In this case, each image is 2 GBs in size. The extraction of a region around a star of interest required the movement of the entire image to a processing platform, which took 4 minutes. A remote proxy was written that supported the image cutout operation directly at the remote storage system, shortening the time for completion to a few seconds.

All collections we support are multi-site. Replication across sites is essential for:

- Disaster recovery. We cannot afford to have a collection lost due to fire or earthquake
- Fault tolerance. When a site is down, we can still access the data from the alternate site.
- Performance. We can load balance accesses across sites
- Curation. Data is managed and maintained by experts that reside at different institutions. The primary copy tends to be at the site where the expertise is located.

13.3 Scenarios

The primary scenario is the execution of the archival processes listed above. The Storage Resource Broker has implemented all of the capabilities listed above, and is in production use in support of multiple persistent archives. They include:

- California Digital Library, crawl of federal web sites, resulting in 16.9 million digital entities, 1.5 TBs of data. The digital entities are registered into the SRB logical name space, and access through a web browser http interface. This makes it possible to display the archived material through the same web mechanisms used to access the original. The URLs for each digital entity are mapped as attributes onto the logical name space used to register the digital entities.

- NARA persistent archive. In this project, the NARA digital holdings are registered into the SRB data grid, replicated between U Md, NARA, and SDSC. Currently over 1.5 TBs of data is registered.
- NSF National Science Digital Library. SDSC runs a persistent archive that holds a copy of each digital entity that is registered into a central repository at Cornell. The number of digital entities is rapidly growing. The system currently has 1.5 million digital entities, with an average size of 50 Kbytes.

13.4 Involved resources

The Persistent Archive contains up to one Peta Byte Data and several dozens million files.

The Storage Resource Broker is installed on:

- Sun, AIX, Linux, 64-bit Linux, HP True-64, Mac OS X, Windows NT

and is used to access:

- File systems (Unix, Mac OS X, Windows, and Linux), archives (HPSS, Unitree, ADSM, and DMF), databases (DB2, Oracle, Sybase, Informix, SqlServer, Postgres), object ring buffers, hierarchical resource managers, web sites, FTP sites.

and provides access to the systems through APIs requested by the application areas:

- C library calls, C++ library calls, Unix shell commands, Python library, Windows DLL library, Windows browser, Web browser, Open Archives Initiative, WSDL, Java

13.5 Functional requirements for OGSA platform

We have the challenge that the preferred access mechanism is specified by the user community. In all cases, they prefer to continue to use legacy APIs for access to distributed data. An example is the CMS high energy physic project at Caltech. They have developed an analysis program called Clarens, which was based on Python. Hence they requested a Python I/O library for interacting with the SRB.

The digital library community (NSF NSDL project) required the use of the Open Archives Initiative protocol for exchanging metadata. This is a simple packaging of the metadata that is exchanged between sites.

The web services description language environment is based on Java. Hence we implemented a pure Java interface to the SRB.

A major distinction between the services provided for current persistent archives and OGSA based persistent archives is the integration of capabilities into composite sets. We are under pressure to optimize the ability to manage bulk registration of files into the logical name space, bulk loading of data onto a storage repository, bulk extraction of data, and bulk deletion of data. This means that we have to issue one request, and then perform operations on 10,000 to 10,000 files. To accomplish this, we do the following:

- Integrate authorization, determination of file location, file access, and file retrieval into a single command. The data grid must process each of these operations without requiring additional interaction with the user.
- Support bulk registration. This is the aggregation of location information about remote files into a series of metadata concatenation files, and the bulk load of the files into the metadata registry. Rates on the order of 600-1000 per second are needed.

- Support bulk loading. This is the combined aggregation of files into containers, and the aggregation of location information into a metadata catalog

A second distinction is the implementation of consistency constraint mechanisms that work across multiple services. Consider access controls on containers that are replicated. In the SRB, the access controls apply to each digital entity that is registered into a container, for all copies of the container. The access controls are a property of the logical name space. Operations on the logical name space result in "completion state" information that is mapped as attributes onto the logical name space and stored in the metadata catalog. To make the problem more specific, consider writes to a file that has been aggregated into a container that was replicated. The data grid needs to implement the following:

- Mapping of access controls onto the logical name space
- Management of write locks on the container
- Management of synchronization flags on the replica copies
- Mechanism to synchronize the replicas

A similar set of constraints emerges when the data is encrypted or compressed. Again the state of encryption/compression needs to be a property of the logical name space, such that no matter where the data is moved, the correct encryption algorithm can be used before transport, and the correct decryption algorithms can be invoked by a client.

The required set of services depends strongly upon the application area. Thus 3D visualization of multi-terabyte data sets requires the ability to do partial file reads, seeks, and paging of data into a 3D renderer. An OGSA service that supports paging of data may be too heavy weight for the 3D rendering system. Services are also needed for data and metadata manipulation. An example of metadata manipulation is the automated extraction of metadata from a file at the remote storage repository, and the bulk load of the metadata into the metadata repository. An example of metadata discovery is the OAI-based metadata extraction, and the formatting of extracted metadata into an HTML or XML file. An interesting metadata service is the provision of access control lists on metadata attributes, as well as on the digital entities.

For data grids, the major challenge is the consistent management of "completion state". For any large collection, the metadata must be maintained in a consistent state with respect to the digital entities. We use databases to manage the state information in "hard state" repositories. Metadata updates are done within the service, internal status information is kept for operations which are in a partial completion state (such as a write to a replica, we need to eventually synchronize across copies).

Explicit data operations include:

- Change permission - Can be used to change access permission on a data grid collection or a data set.
- Copy - Copy contents of data grid collection or a dataset into a new collection or a dataset respectively within the default storage resource or any other storage resource
- Create - Create a new container or a collection
- Ingest data set - Insert a data set present as an attachment to the data grid request
- Download data set - Download a dataset as an attachment to a data grid response
- Delete - Delete a data grid collection or a dataset
- List - List the contents of collection or a container
- Prepare ticket - Prepare a new Grid Ticket

- Rename - Rename a collection or a data set
- Replicate - Replicate the contents of a collection or a dataset
- SeekN'Read - Seek to a point in a data set and read (get) specified bytes as an attachment
- SeekN'Write - Seek to a point in a data set and write (put) the bytes present in the attachment

13.6 OGSA platform services utilization

Utilizing the OGSA data services, the persistent archives will implement bulk registration, load, unload, and delete functions.

13.7 Security considerations

The persistent archive should provide access control for stored data. The current SRB interoperates with GSI 1.1 and GSI 2.4. The next step is to interoperate with GSI 3.

13.8 Performance considerations

The ultimate goals are to use all available bandwidth, register 1000 files per second

13.9 Use case situation analysis

We are not currently using OGSA. Instead we have implemented native APIs and WSDL/SOAP.

13.10 References

[1] R. Moore, A. Merzky, "Persistent Archive Concepts", Global Grid Forum Persistent Archive Research Group, draft on Persistent Archive Recommendations, May 3, 2003.

[2] R. Moore, "Common Consistency Requirements for Data Grids, Digital Libraries, and Persistent Archives", Grid Protocol Architecture Research Group, Global Grid Forum, Tokyo, Japan, March 5, 2003.

[3] R. Moore, C. Baru, "Virtualization Services for Data Grids", Book chapter in "Grid Computing: Making the Global Infrastructure a Reality", John Wiley & Sons Ltd, 2003.

[4] Arcot Rajasekar, Michael Wan, Reagan Moore, George Kremenek, Tom Guptil, "Data Grids, Collections, and Grid Bricks", Proceedings of the 20th IEEE Symposium on Mass Storage Systems and Eleventh Goddard Conference on Mass Storage Systems and Technologies, San Diego, April 2003.

[5] Michael Wan, Arcot Rajasekar, Reagan Moore, Phil Andrews, "A Simple Mass Storage System for the SRB Data Grid", Proceedings of the 20th IEEE Symposium on Mass Storage Systems and Eleventh Goddard Conference on Mass Storage Systems and Technologies, San Diego, April 2003.

[6] Arcot Rajasekar, Michael Wan, Reagan Moore, Arun Jagatheesan, George Kremenek, "Real Experiences with Data Grids - Case studies in using the SRB", International Symposium on High-Performance Computer Architecture, Kyushu, Japan, December, 2002.

[7] R. Moore, "The San Diego Project: Persistent Objects", Proceedings of the Workshop on XML as a Preservation Language, Urbino, Italy, October 2002.

[8] Edward A. Fox, Virginia Tech: Reagan W. Moore, San Diego Supercomputer Center; Ronald L. Larsen, University of Pittsburgh; Sung Hyon Myaeng, Chungnam National University; and Sung-Hyuk Kim, Sookmyung Women's University, Toward a Global Digital Library:

Generalizing US-Korea Collaboration on Digital Libraries, D-Lib Magazine, October 2002, http://www.dlib.org/

[9] Arcot Rajasekar, Reagan Moore, Bertram Ludäscher, Ilya Zaslavsky, "The Grid Adventures: SDSC's Storage Resource Broker and Web Services in Digital Library Applications: 4th Russian Conference on Digital Libraries, Dubna, Russia, October, 2002.

[10] R. Marciano, B. Ludaescher, I. Zaslavsky, R. Moore, and K. Pezzoli, "Multi-level Information Modeling and Preservation of eGOV Data", First International Conference, EGOV 2002, Aix-en-Provence, France, September 3, 2002

[11] G. Bruce Berriman, David Curkendall, John Good, Joseph Jacob, Daniel S. Katz, Mihseh Kong, Serge Monkewitz, Reagan Moore, Thomas Prince, Roy Williams, "An Architecture for Access to a Compute Intensive Image Mosaic Service in the NVO," SPIE Conference 4686 "Virtual Observatories", Hawaii, August 2002.

[12] R. Moore, A. Merzky, "Persistent Archive Basic Components", Persistent Archive Research Group, Global Grid Forum; July 27, 2002

[14] A. Rajasekar, M. Wan, R. Moore, "mySRB and SRB, Components of a Data Grid", 11th High Performance Distributed Computing conference, Edinburgh, Scotland, July 2002.

[15] R. Moore, "Preservation of Data, Information, and Knowledge," Proceedings of the World Library Summit, Singapore, April 2002.

[16] R. Boisvert, P. Tang, "The Architecture of Scientific Software," pp. 273- 284, "Data Management Systems for Scientific Applications," Kluwer Academic Publishers, 2001.

[17] C. Chen, "Global Digital Library Development," pp. 197-204, "Knowledge-based Data Management for Digital Libraries," Tsinghua University Press, 2001.

[18] A. Rajasekar, R. Moore, "Data and Metadata Collections for Scientific Applications", High Performance Computing and Networking (HPCN 2001), Amsterdam, Holland, June 2001.

[19] H. Stockinger, O. Rana, R. Moore, A. Merzky, "Data Management for Grid Environments," European High Performance Computing and Networks Conference, Amsterdam, Holland, June 2001.

[20] R. Moore, "Knowledge-based Grids," Proceedings of the 18th IEEE Symposium on Mass Storage Systems and Ninth Goddard Conference on Mass Storage Systems and Technologies, San Diego, April 2001.

[21] B. Ludäscher, R. Marciano, R. Moore, "Preservation of Digital Data with Self-Validating, Self-Instantiating Knowledge-Based Archives," ACM SIGMOD Record, 30(3), 54-63, 2001.

14 Mutual Authorization

14.1 Summary

The document “Grid Authentication Authorization and Accounting Requirements” is an informational document created by SA3-RG and it describes security requirements for Grid. One of the important requirements mentioned in the document is the need for mutual authorization. This requirement, however, cannot be provided by the current Grid toolkit technology.

When a job is submitted to a specific resource on the Grid, the user is authorizing this resource to run the job and process the resultant data implicitly through the act of targeting this resource for the job submission. However, the specified resource may in turn, transfer or re-submits this job to another resource because of load balancing or satisfying expected quality of service. This secondary remote resource may be trusted by the Virtual Organization, but not by the owner of the Grid job.

14.2 Customers

The mutual authorization requirement comes in general from large site customers with specific security needs to protect the Intellectual Property (IP) of the Grid job or the resultant data. This is the Use Case Commercial Data Center or National Fusion Collaboration but with the added or specific security need.

14.3 Scenarios

This need can be seen in a scenario where the user submits a Grid job which uses or produces sensitive data or the job itself has IP value. The Grid VO may trust a variety of computers but the user may not want this job run on an OS known for security breaches or only on OS with particular OS security features or updates.

14.4 Involved resources

The utilized resources should have a callback service to the user. The callback is used before the resources transfer or re-submit the Grid job to another remote resource. This call back identifies the secondary remote resource and the user’s associated Grid job. The user will handle this mutual authorization call to authorize the secondary remote resource.

14.5 Functional requirements for OGSA platform

The following list include necessary functional requirement of OGSA document for this use case

- Policy
- Multiple Security Infrastructures
- Perimeter Security Solutions.

14.6 OGSA platform services utilization

The following list includes utilized services of OGSA document by mutual authorization use case.

- Name resolution and discovery
- Security
- Policy
- Events

- Service Orchestration.

14.7 Security considerations

The security considerations are described above.

14.8 Performance considerations

The mutual authentication process should be automated and expedient.

14.9 Use case situation analysis

The current use cases do not currently seem able to handle this requirement. OGSA virtualizes the Grid and the resources and computers that comprise the Grid. The requirement for Mutual Authorization requires end to end knowledge of job distribution.

14.10 References

http://www.gridforum.org/2_SEC/SAAA.htm

15 Resource Usage Service (RUS)

15.1 Summary

The Resource Usage Service (RUS) facilitates the mediation of resource usage metrics produced by applications, middleware, operating systems, and physical (compute and network) resources in a distributed, heterogeneous environment. It is one of the core services in the Open Grid Services Architecture.

15.2 Customers

The RUS will be exploited by customers interested in measuring resource consumption for a number of reasons, usually motivated by scenarios related to cost allocation and capacity planning. Potential customers come from both the commercial and scientific domains.

15.3 Scenarios

The RUS is intended to support a wide variety of usage scenarios including those based on: cost allocation (i.e., chargeback); capacity and trend analysis; fraud and intrusion detection; dynamic provisioning; service level agreement compliance; pricing of web services; and workload management.

15.4 Involved resources

Involved resources include all resources whose utilization needs to be measured.

15.5 Functional requirements for OGSA platform

The following list describes the relationship of functions outlined in the Open Grid Services Architecture document to those functions performed by the Resource Usage Service.

- **Discovery and brokering**

  The RUS may use discovery mechanisms to locate resources producing resource usage metrics.

- **Metering and accounting**

  The RUS is a key part of this function.

- **Data sharing**

  No known requirements.

- **Virtual organizations**

  No known requirements.

- **Monitoring**

  The RUS uses function provided by the Monitoring fabric to collect usage metrics.

- **Policy**

  Policy Services will drive the configuration and orchestration of RUS instances.

- **Security**

  Security should support accounting capabilities present in traditional Authentication, Authorization, & Accounting (AAA) systems. Several commercial-based scenarios require the Resource Usage Service to tag consumption metrics with account codes obtained from the AAA system.

15.6 OGSA platform services utilization

The following list describes the relationship of services outlined in the Open Grid Services Architecture document to the RUS.

- **Core Service: Name resolution and discovery**

  The RUS will use this core service for resolving handles into references.

- **Core Service: Service domains**

  The RUS will probably not use this service.

- **Core Service: Security**

  The RUS will use AAA function to obtain account codes. Also, Security services will be needed to protect against unauthorized access to resource usage metrics. Authorization control is required for both operation invocation and access to service data elements.

- **Core Service: Policy**

  RUS instances will be configured using the Policy service.

- **Data and Information Services: Data Management**

  The RUS will probably not use this service.

- **Data and Information Services: Messaging, queuing, and logging**

  The RUS exchanges metrics using messaging and queuing. RUS requires Logging services for audit and recovery.

- **Data and Information Services: Events**

  Resource Metrics are events and the RUS should exploit and conform to the Event services.

- **Data and Information Services: Metering and accounting**

  The RUS is a member of this set of services.

- **Data and Information Services: Transactions**

  The RUS will probably not use this service.

- **Management of Computation and Resources: Service Orchestration**

  Since resource usage is metered in a distributed environment, RUS instances need to be wired together (orchestrated) with other infrastructure (e.g. messaging) services.

- **Management of Computation and Resources: Administration**

  The Administration service manages the deployment, changes, and identity of RUS.

- **Management of Computation and Resources: Provisioning and resource management**

  Provisioning systems use resource usage metrics obtained from the RUS to make their provisioning decisions.

- **Management of Computation and Resources: Reservation and scheduling services**

  The RUS will probably not use this service.

- **Management of Computation and Resources: Deployment services**

  The Deployment service will be used to deploy the software that supports the RUS.

15.7 Security considerations

RUS requires security services to protect access to potentially sensitive resource usage information. Also, as the RUS will exploit accounting information extracted from the AAA system.

15.8 Performance considerations

To minimize the cost of accumulating resource usage data, the implementations of the RUS must be very efficient. In general, the cost of measuring resource consumption should be a small fraction of cost of total resource consumption.

15.9 Use case situation analysis

Since RUS consumes metrics generated by underlying resources. There appears to be a need for standard semantics and policy for controlling resource instrumentation. Perhaps this function should be covered in the Web Service Distributed Management (WSDM) or Common Management Model (CMM).

15.10 References

http://www.ggf.org/3_SRM/rus.htm

16 Editor Information

Ian Foster, Distributed Systems Laboratory, Mathematics and Computer Science Division
Argonne National Laboratory, Argonne, IL 60439
Phone: 630-252-4619, Email: foster@mcs.anl.gov

Dennis Gannon, Indiana University, Bloomington, IN 47405
Phone: 812-855-5184, Email: gannon@cs.indiana.edu

Hiro Kishimoto, Grid Computing & Bioinformatics Laboratory
Fujitsu Laboratories Limited, Kawasaki, Japan 211-8588
Phone: +81-44-754-2628, Email: hiro.kishimoto@jp.fujitsu.com

Jeffrin J. Von Reich, Software Engineering Business Unit,
Hewlett Packard, Fort Collins, CO, USA – 80528
Phone: +1-9708980700, Email: jeffrin_von-reich@hp.com

Acknowledgements

This work was supported in part by IBM; by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy under Contract W-31-109-Eng-38 and DE-AC03-76SF0098; by the National Science Foundation; and by the NASA Information Power Grid project; and by Hewlett Packard.

We gratefully acknowledge the contributions made to this document by Nedim Alpdemir, Takuya Araki, Boas Betzler, Kate Keahey, William Lee, Tan Lu, Norman Paton, Jon MacLaren, Andreas Savva, Charles Severance, David Snelling, Ravi Subramaniam, Fred Maciel, Takuya Mori, Andrew Grimshaw, Jem Treadwell, Latha Srinivasan, GGF Working groups and all other members of OGSA working group members who discussed and reviewed the material.

References

Each chapter has reference section.