GWD: draft-ggf-gesa-use-cases-02
Grid Economic Services Architecture WG

S. Newhouse
J.MacLaren
K. Keahey

GESA Use Cases

Document: sched-gesa-1.0
Category: Informational
Date of this version: 6th of June 2003

Status of this Draft

This document provides information to the community regarding use case scenarios occurring in the context of Grid economics.

This document is a product of the Grid Economic Services Architecture (GESA) Working Group of the Global Grid Forum.

Distribution of this document is unlimited.

Copyright

Please see Section 6 for full Copyright statement.

Abstract

This document surveys use cases occurring in the context of Grid economics. Specifically we describe and analyze use cases of computational provider, application service provider, software application provider, brokering service provider, and computational reseller. The intent of this document is to provide analysis of these use cases, and lead to articulating requirements for a Grid Economic Service Specification. These use cases will determine the features we want to cover in our architecture approach.

Table of Contents

1 Introduction

This document contains a description and analysis of GESA use cases. Specifically, we describe the following use cases:

- Computational Provider (providing primarily hardware resources)
- Application Service Provider (providing primarily execution: software and hardware on which the software executes)
- Software Application Provider (providing primarily software)
- Brokering Service Provider (a reseller: matching consumers to providers)
- Computational Reseller (reselling resources)

We analyze the use cases and make recommendations about the economic capabilities they motivate.

2 Use Cases

2.1 Computational Provider Use Case

This use case was contributed by Jon MacLaren, Steven Newhouse, Thomasz Haupt.

2.1.1 Summary

There are many instances where a computer user finds locally available computational resources inadequate for their requirements. Currently, their main option is to try and locate a suitable resource, obtain an account there, learn about using this new resource, and finally submit their work to the resource; their only other option is to make do with what they have! Worse still, obtaining an account may not be possible due to bureaucratic hurdles.

2.1.2 Scenario(s)

A user with a computational job to do cannot satisfy their requirements with locally available resources. They take these requirements to a resource broker (e.g. architectural, installed applications, cost of cycles and turnaround time), using which, they discover a computational resource able to satisfy their requirements.

The user does not have an account at this site, but is able to find a suitable method to pay for the work to be done. In addition, the new site finds the security credentials of the prospective user to be acceptable; similarly, the user finds the site's credentials to be acceptable.

The user and the site make any final negotiations, finding an agreed turnaround time and cost, resulting in a form of contract between them. The user accepts this contract by submitting the work.

Following the successful delivery of the resources, payment, which was secured earlier, is finally processed. The user also collects the results of the computation.

2.1.3 Example Usage

There are many rare and/or expensive computing resources which currently lie outside the grasp of potential users, e.g. supercomputers, vector processing machines. Obtaining access to such resources is generally complicated for members of communities which can gain access. For the casual user, i.e. the "nerd in the street", access is typically impossible.

Even though such computers are not very usable by everyday programmers, I would argue that the following groups would make use of such a facility:

- Academics with small supercomputers (16 processors), who want to run their codes on a large machine of the same architecture, to study scalability, or for obtaining points on a speedup curve for a paper.

- Non-academic companies may have sufficient compute capacity to handle their workload for 90% of the year, but require additional computing power near the end of the financial year.

- Scientists with codes that perform well only on a vector processing machine, and who cannot find such resources locally.

- The trusted computational reseller, who has a significant customer base (or is often approached by brokers) keen to obtain such resources; this results in the sale of a large chunk of resources for use in the future.

- The "Nerd-in-the-street" who would like to make use of a large supercomputer as a one-off.

Such usage patterns cannot be supported by current registration and accounting systems.

2.1.4 Commercial Perspective

The buyer is paying for their computational work to be done, i.e. purchasing computational power which can satisfy his/her requirements. Some amount of memory and temporary disk storage will also be used at the same time. There may also be the purchase of a single-use licence for a particular application, or funds to "rent" a site-licence (or part thereof) for the duration of the job.

The seller is providing time on some computational resource, i.e. CPU cycles on some machine or cluster. The seller may also be wishing to offset some of the cost of expensive software licences that have been purchased by renting the use of the licensed application. Alternatively, if a pay-per-use license is available, the site owner may be taking a cut on each use of the software.

Conversely, this can also be viewed as the seller bidding for work. This is especially applicable in the case of an underloaded machine, where the owner is trying to prevent potentially valuable cycles from being lost.

Naturally, all costs may depend on the user's status, e.g. academic vs. commercial user.

2.1.5 Consumed Resources

CPU cycles on a machine or cluster are being consumed. Presumably, some amount of disk space, and memory are also being used. Some sort of pay-per-use licence could be being traded as well, or a permanent site-licence (or part thereof) could be being "rented out".

2.1.6 Architectural Requirements

Only requirement relevant to GESA will be described here.

For the computational provider to be able to reach as large a potential market as possible, they must be able to accept as many payment mechanisms as possible. Therefore, the architecture must be agnostic in these terms, e.g. allowing a Visa transaction, or invoice and purchase order systems, etc.

As such transactions could be many and frequent, it should also be possible for payments to be aggregated, e.g. for monthly billing. (If the centre does not want to handle this sort of accounting, they should be selling large chunks of resource to resellers – see Computational Reseller Scenario.)

When the customer is a one-off or ad-hoc user, the purchase is likely to be for a small amount of resource in the near future. When the customer is a reseller, the purchase is likely to be larger, and further into the future. Therefore the architecture must be able to handle varying granularities of requests. Also, to make the selling process more flexible, and therefore more attractive to the reseller, it should be possible to purchase of resources where the time of use is not precisely stated, e.g. "I will buy 512 CPU hours on green.cfs.ac.uk for consumption some-time over the following month (which I will re-sell)"

2.2 Application Service Provider

This use case has been contributed by Kate Keahey.

2.2.1 Summary

The service provider is providing access to service execution (comprising software and resources as opposed to just one of them). Certain quality-of-service (QoS) constraints may be requested by the client. Likewise, the service provider may make representations about provided quality of service in terms of execution time, response time, accuracy of results, etc. The charge for the service may be based on the QoS provided (faster execution costs more) or reliability with which

this QoS has been provided. Note, that the eventual QoS exposed by the application service must be an end-to-end QoS from the point of view of the client.

The contact describing the constraints associated with the service can be expressed as a service level agreement (SLA) [].

2.2.2 Scenario(s)

The National Fusion Collaboratory (NFC: www.fusiongrid.org) provides network services allowing for better integration of experiment, analysis and simulation. Fusion experiments are run in a pulsed more with new experimental pulses taking place every 15-20 minutes. The analysis of each pulse could serve as input to the next pulse. In this situation, it is important to be able to perform the analysis of a pulse within a roughly 10 minute time window in order to allow sufficient time for digesting the results, and adjusting the experimental parameters accordingly. Up to now, the prevalent way of dealing with this problem was to buy more and more dedicated resources. An alternative is to use remote computational resources, which would be possible if only their execution could bounded by the requisite real-time parameters. At the same time, less time-critical executions could be provided at a lesser price.

2.2.3 Example Usage

[go through steps]

2.2.4 Commercial Perspective

The seller is providing application development and maintenance on a familiar set of resources (platforms). In addition, the seller is also providing (if necessary exclusive or pre-emptive) usage of resources necessary to provide end-to-end QoS: execute all the codes involved, move and store data, etc. as well as orchestration of codes and resources. In some cases (such as NFC) the service provider may provide additional capabilities such as on-the-fly diagnostics and debugging of a service. Finally, the server makes representations about the QoS that can be obtained from a service which could include predictions etc. (in other words, the seller also sells the ability to estimate/predict certain qualities).

The buyer buys execution with a certain QoS: execution time, response time on a query, security of interaction, or reliability of execution.

Charges could be made based on requested/delivered QoS (based on the QoS itself and/or reliability with which it was delivered), the resource usage incurred during the actual execution. In addition, charges could be made based on a monthly subscription (say to a database service), accumulation of small charges, or one time per-execution charge. They also could be made based on a “Grid session” for a user or a virtual organization, or on a per-service usage. [although these remarks emerged in the ASP context, they really seem more suited to more than just this use case]

2.2.5 Consumed Resources

The consumed resources in this case include code development, maintenance, and execution, resource usage, orchestration services (hopefully general-purpose software itself combining codes and resources), prediction and reservation services (and other services necessary to issue the SLA).

2.2.6 Architectural Requirements

The enabling technologies include: software supporting making statements about the provided QoS, contract management software, the ability to transfer contract execution rights between buyers and sellers.

2.3 Software Application Provider

This use case was contributed by Steven Newhouse.

2.3.1 Summary

The current model of using a software library (or program) provided by a commercial vendor is for that library to be installed on a specific host. The library is enabled, following an exchange of money and the acceptance of the terms of the licence agreement by the host organisation, with a licence key for a particular combination of host and platform. The amount paid for the licence may be dependent on the platform, the user (e.g. academic/commercial), the number of concurrent users, the duration of the licence, the type of support, the number of users and the functionality within the library.

Within the grid a user may have a variety of machines available for their use. Not all of these machines will have the necessary software required for all user activity. Nor will the system managers wish to pay for and install software for casual remote users. Therefore the ability to 'buy, install & execute' a software library or application on demand is essential.

2.3.2 Scenario(s)

A user wishes to conduct a fluid dynamics analysis using their favourite CFD software. They establish which computational resource they will be using [Link to other use case?] and wish to dynamically install the CFD software onto that resource. By providing information on the execution host and the expected duration of the analysis they request a licence and the binary. A licence key is provided with the downloaded software library for unpacking and use on the machine. The purchase of the licence key (start & end date) needs to be coordinated with the reservation on the computational resource.

2.3.3 Example Usage

[Describe the process through a series of concrete examples that demonstrate the use case.]

2.3.4 Commercial Perspective

The seller is providing access to their intellectual property encapsulated with a software library or application. This is exposed to a user on their machines through a well-defined programmatic API that may be dynamically linked into the users own library. The buyer is obtaining access to

this functionality as and when they want it on the resource that they want. Currently, the installation and configuration of these libraries is a privileged operation due to the provisioning of the licence keys.

2.3.5 Consumed Resources

The focus in this use case is to enable the invocations on a software API. This may be licensed on a per invocation basis [Is this possible with say FlexLM?] or over a (short) time period. The items being charged will therefore be a (sub)set of a library'sAPI for a particular duration on a specific host.

2.3.6 Architectural Requirements

The software provider needs to provide information on the availability of the software (which versions on which platforms and any requirements for dependent libraries, e.g. MPI for commercial ScaLAPACK library).

The primary constraints on the architecture are the ability to pay for the service.

Current pricing models may be very dependent on the type of user (e.g. commercial or academic), their location (e.g. USA, Europe, etc) or the level of support offered (e.g. none, telephone, application consultancy). These pricing models should be supported in the GESA model although there is an argument that in the long-term a pricing model may be user agnostic.

2.4 Brokering Service Provider

Author(s): Kate Keahey (keahey@mcs.anl.gov)

2.4.1 Summary

The service provider is providing a service as a "reseller" matching different client request to different provider offerings.

2.4.2 Scenario(s)

Resource Broker. A user seeks a resource with described by a certain set of parameters (for example: availability of networking infrastructure, certain number of CPUs of defined power, proximity of storage resources). The brokering service presents the user with a list of acceptable (or near-acceptable) choices and iterates with the user on refining the set of parameters. The user eventually settles on a resource. The broker is paid a "commission fee" (representing for example a percentage of what the user pays to the resource provider) paid by either the buyer or a seller, or a flat service fee.

Other brokers include information broker, application service provider broker etc.

2.4.3 Example Usage

[usage scenario]

2.4.4 Commercial Perspective

The seller is primarily a reseller, as it functions as a "middleman" selling services provided by others. In that sense, the seller sells mainly information. In some cases, the seller may resell retail resources acquired by "wholesale" sale. Also, the reseller may specialize in "combination sales", that is acquire separately multiple resources (disk, network, CPU reservations) and sell them as a "bundle".

The buyer is buying information, convenience, "complex products" (a combination of resources required for a particular task). Also, the buyer may be buying a warranty for the specific resource combinations.

The charges may be as a flat fee or as a percentage of transaction with the original seller.

2.4.5 Consumed Resources

The resources involve mainly maintaining databases of relevant information, may also involve co-scheduling (or co-reserving) access to multiple resources at once.

2.4.6 Architectural Requirements

The requirement is the capability to accurately describe resources, accurately describe requirements for resources and to provide capabilities such as reservations in terms of service level agreements (SLAs).

2.5 Computational Reseller Use Case

This use case was contributed by Jon MacLaren and William Lee.

2.5.1 Summary

It is not always desirable for computational resource provider to interface with consumers directly. A supply chain between the providers and consumers allow resource provider to concentrate on their core competence in maintaining large compute resources and avoid providing costly interaction and care to a large number of consumers. End users can purchase resources bundled into attractive packages, possibly coming from more than one upstream provider. The resellers can make money from reselling aggregated computational resources without having to own any resource themselves, thereby minimising their own risk as well as their consumers by sourcing multiple providers. It allows them to focus in providing good customer care as well as optimising resource bundles for their target market.

2.5.2 Scenario(s)

A computational provider, preferring not to deal direct with customers, sells time on their computational resource to a reseller. The reseller markets the resource to downstream

consumers, bundling the resource with other resources, sourcing multiple upstream providers. Consumer purchases the resources in a monthly package which includes items such as CPU cycles, secure disk storage, database and software licences. The reseller maintains the agreed bundle by sustaining their relationships with upstream providers, or possibly switch provider without affecting the agreed service level with the end users.

2.5.3 Example Usage

Consider the example of a reseller who has strong relationship with the chemical industry and the expertise to deal with chemistry applications running on supercomputers. Their upstream providers are supercomputer centres. Their downstream consumers are chemists who want to use these applications. It is useful to consider this example in three perspectives.

1. **The upstream resource provider.** The supercomputing centre wishes to sell its cycles to a reseller because either:

   a) During a period of low usage, to sell large amounts of redundant cycles (or buy work); or

   b) During normal usage, the resource owner never needs to deal directly with large number of small customer group (costly to maintain high quality of customer care. Its own advantage!). Further, this policy enables them to manage their resources in a small number of large transactions.

   c) For future period that expects high / low demand, sell a small / large portion of the cycles in advance to maintain a steady usage at a reasonable price.

2. **The reseller themselves.** The reseller bundles the resources available to it from the upstream providers, plus some licences it can obtain from the software vendors at a reduced rate (as it only deals 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 Guassian98 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). Reseller who has insights in market trend can predict future demand and source resource provision from upstream vendors in advance when price is attractive.

3. **The downstream resource consumer.** 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.

2.5.4 Commercial Perspective

[The computational provider is selling resources to the reseller – see Computational Provider Scenario.]

The reseller is buying compute resources from an upstream computational provider.

The reseller bundles sources from multiple suppliers, and may bundle the computational resource with other resources such as licences and offline disk storage, which they then sell to downstream consumers.

[The downstream consumers then buy these attractive bundles.]

2.5.5 Consumed Resources

Ultimately, it is the resources bought from the providers that are being consumed by the end users. However, the charging is more complex due to the presence of one or more resellers in the supply chain.

2.5.6 Architectural Requirements

Reselling must be permitted (but only if desired). The resource provider must be able to retain some amount of control over who the end-users are (and resellers further down the chain). This means that some form of policy is required and contracts whereby their resources can be reclaimed if the reseller violates this policy.

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

There must be some support for aggregating resources (from OGSA, I suppose), and therefore somehow aggregating all the policies of upstream providers (our responsibility, I think).

3 Use Case Summary

4 Analysis

1.

5 Contact Information

Kate Keahey
keahey@mcs.anl.gov

Jon MacLaren
jon.maclaren@man.ac.uk

Steven Newhouse
s.newhouse@doc.ic.ac.uk

6 Copyright Notice

7 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.