
Name of use case: Visualization Service for the Grid

Contact: Pascal Kleijer [k-pasukaru@ap.jp.nec.com]
 Eiichi Nakano [e-nakano@ax.jp.nec.com]
 NEC, HPC Marketing Promotion Division

Arihiro Yoshida [yoshida@grid.nii.ac.jp]
 NII, National Institute of Informatics

Authors: Pascal Kleijer

1. General Information:

This section consists of check-boxes to provide some context in which to evaluate the use case.

1.1 Which best describes your organisation:

Industry [X]
 Academic [X]
 Other []
 Please specify:

1.2 Application area:

Astronomy []
 Particle physics []
 Bio-informatics []
 Environmental Sc. []
 Image analysis []
 Other [X]
 Please specify: Massive Data Visualization:
 - Real-time
 - Post-Processing
 - Database Post Visualization Access

1.3 Which of the following apply to or best describe this use case
 Multiple selections are possible, please prioritize with numbers
 from 1 (low) to 5 (high):

Database []
 Remote steering [4]
 Visualization [5]
 Security [1]
 Resource discovery []
 Resource scheduling []
 Workflow []
 Data movement [3]
 High Throughput Computing [1]
 High Performance Computing [5]
 Other []
 Please specify:

1.4 Are you an:

Application user	[X]
Application developer	[X]
System administrator	[]
Service developer	[X]
Computer science researcher	[]
Other	[]
Please specify:

2. Introduction:

2.1 Provide a paragraph introduction to your use case. Background to the project is another alternative. (E.g. 100 words).

This project lies within the NAREGI (National Research Grid Initiative) framework. Our goal is to provide a Grid Service Framework for Massive Data Visualization of various simulations' types, including coupled simulations.

Most simulations in any scientific fields can never accomplish their purpose without visualization (in concurrent or in post-processing), which enables researchers to observe and analyze their unrecognizable numerical results.

This project ambitions the realization of an API for the grid services infrastructure. This API serves to connect two remote systems client and server by the grid medium and offers the client a set of tools to perform visualization of large-scale simulations. The visualization is based on images transferred from the server to the client and visualization controls sent by the client to the server.

Additional features such as Tracking/Steering is also in the possible range of inclusion.

2.2 Is there a URL with more information about the project ?

<http://www.naregi.org>
<http://www.cs.vu.nl/ggf/apps-rg/meetings/ggf12/kleijer-final.pdf>
<http://www.cs.vu.nl/ggf/apps-rg/meetings/ggf12/kleijer-slides.pdf>

3. Use Case to Motivate Functionality Within a Simple API:

Provide a scenario description to explain customers' needs. E.g. "move a file from A to B," "start a job."

Please include figures if possible.

If your use case requires multiple components of functionality, please provide separate descriptions for each component, bullet points of 50 words per functionality are acceptable.

To help understand the following points a small glossary is added:

Server: A visualization capable solver, data-mining program, third party application, etc. running on a grid resource.

Service provider: A grid service or any program that interface the client and server. The provider can run on a different resource

as both server and client, it can also be part of the server application.

Client: The visualization application running on the user's local computer.

The functionalities can be applied to one or more of the following target categories: {common, post-processing, concurrent, database}.

- Locate a resource to host a simulation (with visualization capability). This should be done automatically based on the nature of the simulation the user is preparing. [common]
- Launch the simulation server, including visualization service, on the resource allocated. [common]
- Send visualization control commands and receive a sequence of images back (streaming capability). [common]
- Replay a sequence of image (eventually video) stored on a remote resource. No simulation is running, only post-processing output is read. [post-processing, database]
- Steer the simulation/application (pause, stop, resume, etc). [concurrent]
- Plug-in or Plug-out from a running simulation without affecting it. [concurrent]
- Real-time tracking of simulation parameters. [concurrent]
- Database access to retrieve a stored result. [database]
- File transfer. [common]

4. Customers:

Describe customers of this use case and their needs. In particular, where and how the use case occurs "in nature" and for whom it occurs.
E.g. max 40 words

We can categorize different customer types:

- Post-Processing: They want to visualize and manipulate massive amount of data on site to avoid transfer between systems.
- Concurrent: They want to visualize an on going massive data computation. No need for data output, the simulation results is displayed on the fly.
- Repository Access: They want to access images, videos, or processed simulation results in a repository (database, file system, etc.)
- Application Service: Some applications like PyMol can be made available on high performance systems to perform visualization jobs remotely.

In each case, they are looking toward the Grid for its potential without the hassle of knowing how to operate of the Grid.

5. Involved Resources:

5.1 List all the resources needed: e.g. what hardware, data, software might be involved.

Hardware: High-end computers for simulations and visualization
The hardware does not need High-end rendering capability like a SGI machine. Any HPC system is acceptable.
Client can run on any piece of hardware.

Data: No specific data requirements

Software: Now Unicore and/or Globus as middleware. Later, any OGSA or WSRF implementation will do.

5.2 Are these resources geographically distributed?

Yes. The resources can span from a local machine to a fully earth's spread system.

5.3 How many resources are involved in the use case? E.g. how many remote tasks are executing at the same time?

From 1 to N, where N is the maximum capacity the grid can offer. In the case of coupled simulations the interactions between the different tasks can be quite complex.

5.4 Describe your codes and tools: what sort of license is available, e.g. open or closed source license; what sort of third party tools and libraries do you use, and what is their availability; do you regularly work from source code, or use pre-compiled applications; what languages are your applications developed in (if relevant), e.g. Fortran, C, C++, Java, Perl, or Python.

All code produced within the project will be open-sourced at the finalization of the project. All third party code used must have similar license agreements.

Languages: FORTRAN & C for the visualization side. Java for the client side.

5.5 What information sources do you require, e.g. certificate authorities, or registries.

A registry is required that allows the components of the system to find each other. We would like to use registries such as GIIS or UDDI for resource discovery, but own registry usage is also possible. This will depend on the deployment environment.

5.6 Do you use any resources other than traditional compute or data resources, e.g. telescopes, microscopes, medical imaging instruments.

In general case no, but that may depend on the application developed.

5.7 Please link all the above back to the functionalities described in the use case section where possible.

N/A

5.8 How often is your application used on the grid or grid-like systems?

- Exclusively
- Often (say 50-50)
- Occasionally on the grid, but mostly stand-alone
- Not at all yet, but the plan is to.

The goal of this project is to offer the Grid capability to other applications or projects. We are therefore completely committed to the Grid or Grid-like systems.

6. Environment:

Provide a description of the environment your scenario runs in, for example the languages used, the tool-sets used, and the user environments (e.g. shell, scripting language, or portal).

Simulation servers are generally written in FORTRAN or C. The visualization library has bindings in FORTRAN and C. This code runs, for example, on OSs like Super-UX, AIX, or *NIX systems.

The Grid Service wrappers can be written in Java, C/C++ depending on the application wrapped. Most likely target will be the same as the simulation/visualization servers.

To avoid the hassle of cross-platform compatibility the client sides are targeted to be all Java based.

7. How the resources are selected:

7.1 Which resources are selected by users, which are inherent in the application, and which are chosen by system administrators, or by other means? E.g. who is specifying the architecture and memory to run the remote tasks?

The resources are decided by the problem the user wants to solve. The application will request the registry. The user may have to choose between different available resources if they are available.

7.2 How are the resources selected? E.g. by OS, by CPU power, by memory, don't care, by cost, frequency of availability of information, size of datasets?

OS, Architecture, Memory, disk space, CPU power, software availability (if the service is pre-installed).

7.3 Are the resource requirements dynamic or static?

For a given request the resource is in general static, but this also depends on how the server side is setup.

8. Security Considerations:

8.1 What things are sensitive in this scenario: executable code, data, computer hardware? I.e. at what level are security measures used to determine access, if any?

The information transmitted on the grid by each peer is on most cases highly sensitive. The middleware (communication layer) must

ensure tight security.

8.2 Do you have any existing security framework, e.g. Kerberos 5, Unicore, GSI, SSH, smartcards?

For the moment we exclusively relay on the UNICORE or Globus kits.

8.3 What are your security needs: authentication, authorisation, message protection, data protection, anonymisation, audit trail, or others?

Setup phase: authentication, authorization
Runtime: message protection

8.4 What are the most important issues which would simplify your security solution? Simple API, simple deployment, integration with commodity technologies.

Middleware should take care of the security issues. The application should just specify that high security connection is necessary.

9. Scalability:

What are the things which are important to scalability and to what scale - compute resources, data, networks ?

Large-scale simulations are generally limited by the computation resource available. The scaling would take place on the number of CPU allocated. Another scalable factor would be the image size to control the data size.

10. Performance Considerations:

Explain any relevant performance considerations of the use case.

The computational resource is the most critical issue. In visualization we must expect at least 1 frame/second for rendering (without the simulation computation time, if any)

The second important matter is the network latency, the faster the response the better. Network capacity is dependent on the resolution of the image the user wants, not on the size of the problem to simulate.

11. Grid Technologies currently used:

If you are currently using or developing this scenario, which grid technologies are you using or considering?

Using: WSRF. XML, Unicore, Globus.

Future: SAOP, GridFTP, MPICH-G

12. What Would You Like an API to Look Like?

Suggest some functions and their prototypes which you would like in an API which would support your scenario.

We are currently defining a general visualization framework API. From the client point of view the following code can be considered as an example:

```
Post-Processing Service:
*****
// Generate an instance of Post visualization service.
VisualizationService pvs
    = VisualizationServiceFactory.newInstance(POST_PROCESSING_SERVICE, ...
);

// Set data file which contents is visualized
pvs.setDataFile("gsiftp://localhost:2811/tmp/file1");

// Setting parameter
pvs.setCamera(...);
pvs.setLight(...);

// Generate a visualized image
byte[] imageData= pvs.getImage();

Concurrent Service:
*****
// Generate an instance of Concurrent service.
VisualizationService cvs
    = VisualizationServiceFactory.newInstance(CONCURRENT_SERVICE, ...);

// Setting initial parameter
cvs.setCamera(...);
cvs.setLight(...);
cvs.setBackgroundColor(...);
cvs.setImageSize(...);

// Get the visualization streaming. Images or image sections are
// streamed on an input stream.
InputStream vizInput= cvs.getInputStream();

// Visualize
while (...)
{
    // Process the content of the stream.
    ...

    // Set new parameters.
    ...
}
```

In these examples we consider that authentication, job launching, etc is done prior to get a visualization service.

13. References:

List references for further reading.

- a) Pascal Kleijer, Eiichi Nakano, Toshifumi Takei, Hiroshi Takahara,
Arihiro Yoshida
API for Grid Based Visualization Systems

GGF 12 Workshop on Grid Application Programming Interfaces.

- b) <http://www.naregi.org>
