

# API for Grid Based Visualization Systems

Pascal Kleijer<sup>\*</sup>, Eiichi Nakano<sup>\*</sup>, Toshifumi Takei<sup>\*</sup>, Hiroshi Takahara<sup>\*</sup>,  
Arihiro Yoshida<sup>+</sup>

*<sup>\*</sup>NEC Corporation, HPC Marketing Promotion Division*

*<sup>+</sup>NII, National Institute of Informatics*

In this paper, the authors discuss their proposal for a Grid Based Visualization API (Application Program Interface). Grid enabled system are becoming more popular with the years and therefore great efforts are put in it to develop new functionalities. Visualization of large-scale distributed data is a necessity for scientific simulation. The confluence of both ideas is therefore natural. We propose two different APIs depending on the level in the workflow connectivity: Visualization Library API and a Grid Visualization Service API. In the paper, both APIs usages are detailed and practical use cases are provided.

## 1 Introduction

Grid computing has become a major topic of discussion, research and development since the turn of the millennium. Many groups, associations and conferences are setup to speak, discuss and interact on the topic. Major investment is done in developing grid hardware and middleware. The focus is set on the grid allocated resources and how to interconnect them. At its core, grid computing is based on an open set of standards and protocols – e.g., Open Grid Services Architecture (OGSA) – that enable communication across heterogeneous, geographically dispersed environments. Globus [1] and UNICORE [2] are both the major implementations available now-a-days for Grid middleware deployments.

However, since short, a new area has seized part of the grid community: grid-computing applications. Without the application layer, the grid computing will have no sense and just be a nice proof of concept. With the focus shifting toward applications, new problems emerge that are not or were not taking in account during the initial development of the core. The requirements may highly differ between each application, but it can always be resumed to a grid-based communication with the middleware medium.

The scientific community is since always greedy of simulations. Simulations are most of the time targeted at High Performance Computing systems (HPC), which is exactly what the grid is tackling and proposing. With the grid it is possible to launch large-scale simulations over a virtual environment. Using this power of the grid to just launch or transfer end results is not satisfactory, more is possible.

1. 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.
2. A solver has not to be static; it can be steered by a simple client on the fly. This enables the researcher to change, alter or rectify the simulation without having to stop and restart it.

NAREGI [14] (National Research Grid Initiative) is one of the collaboration projects among industry, academia, and government, initiated by the Ministry of Education, Sports, Culture, Science, and Technology (MEXT), Japan. Within the NAREGI framework our goal is to providing Grid Service Framework for Visualization of various types of simulations, including coupled simulations. We can resume the major points to:

- Implementation of visualization service interfaces. Making available the use of various visualization tools with an interface compliant with the OGSA.
- Integrated visualization through coordinated visualization services for large-scale distributed data. High-performance visualization hiding the movement of numerical data files.
- Application to user's own coupling simulation programs, such as MO<sup>1</sup>/MD<sup>2</sup>-coupled nano-science applications.

To achieve this goal we have two different API proposals. The first API enables to connect a solver to a visualization service. This API offers the first level of connectivity between a solver and the grid/network/program. The second API uses the grid services infrastructure. This API serves to connect two remote systems client and server by the grid medium. In a first phase, concepts validation and test feasibility will focus on the parallel visualization module because of its high performance capability and already available technology.

## 2 Related Work

### 2.1 RealityGrid

RealityGrid [4] is a consortium of universities and collaborating institutions under the U.K. government's initiative on e-Science [5]. The project aims to grid-enable the realistic modeling and simulation of complex condensed matter structures at the meso- and nano-scale levels as well as the discovery of new materials. The project also involves applications in bioinformatics and its long-term ambition is to provide generic technology for grid based scientific, medical and commercial activities.

The project proposes to extend the concept of a Virtual Reality centre across the grid and links it to massive computational resources at high performance computing centers and experimental facilities by using grid technology to closely couple high throughput experimentation and visualization. The current implementation has a visualization and steering model specific to the problem. It is based on the OGS (Open Grid Services Infrastructure) and uses existing applications linked by a middleware Grid infrastructure.

### 2.2 GEMSS

The GEMSS project [7] stands for Grid-Enable Medical Simulation Service and is a European Union research project. The project is concerned with the creation of medical Grid service prototype in a secure service-oriented infrastructure for distributed on demand supercomputing. The project framework hides the complexity of transforming

---

<sup>1</sup> MO stands for Molecular Orbital: an orbit resulting from overlap and mixing of atomic orbitals on different atoms. An MO belongs to the molecule as a whole.

<sup>2</sup> MD stands for Molecular Dynamics: study of atomic and molecular motion and dynamics at the level of classical dynamics as well as quantum mechanics.

existing applications into Grid services. The visualization is only a small component of the project.

The project implements middleware to interface non-Grid applications with a Grid service. The merit of such approach is that no new application is necessary for the task assigned; only a glue middleware application is necessary to interface each application. The cons are that the coupling middleware application can become very complex with the number of applications to interface. Also it doesn't offer any flexibility with visualization.

### **2.3 UniGrids**

The UniGrids project [8] will develop a Grid Service infrastructure compliant with the Open Grid Service Architecture (OGSA). It is based on the UNICORE Grid software [2] initially developed in the German UNICORE and UNICORE Plus projects.

One of the project's work packages consists of integrating the VISIT toolkit [9]. VISIT (Visualization Interface Toolkit) is a library that supports the development of interactive simulations. It provides functions for establishing a connection between a simulation and visualization, exchanging data and eventually shutting down the connection again. VISIT is developed in the Central Institute for Applied Mathematics at the Research Centre Jülich [10].

### **2.4 TeraGrid**

TeraGrid [13] is a multi-year effort to build and deploy the world's largest, fastest, distributed infrastructure for open scientific research. When completed, the TeraGrid will include 20 teraflops of computing power distributed at five sites, facilities capable of managing and storing nearly 1 petabyte of data, high-resolution visualization environments, and toolkits for grid computing. These components will be tightly integrated and connected through a network that will operate at 40 gigabits per second—the fastest research network on the planet.

This project mainly focuses on hardware and middleware integration. However the software layer do focus on visualization with existing tools. The integration of existing visualization and scientific analysis applications to the environment is done on case-by-case bases and does not seek to create an integrated framework.

## **3 Needs for Server-Side Visualization Service**

An efficient approach is required for visualizing an enormous amount of distributed data produced by parallel simulations that are conducted in the grid computing environments. Conventionally, visualization refers to post-processing simulation results on a user's terminal after the numerical simulation has been completed. The simulation results have to be output to the computational server's disk and transferred to the user's terminal before they are processed. Problems appear when visualizing a large volume of data with the conventional post-processing. For one thing, a large volume of distributed numerical data has to be combined into one batch of organized data and then has to be transferred over the network. Moreover, a large amount of disk space is necessary to store the data somewhere in the grid computing environment and on the user's terminal, and a large amount of memory space is also necessary to manipulate the data on the user's terminal. One solution to such problems is server-side visualization where all of the visualization processes are conducted by utilizing the computing server's resources. Especially, by conducting concurrent visualization on the server side, the storage and

transfer of data resulting from number crunching can be minimized, thus making possible efficient visualization that is tailored to high-end computing.

One of the systems that achieve complete server-side concurrent visualization and post-processing is RVSLIB [15] (Real-Time Visual Simulation Library) that has been developed by NEC [12]. The software conducts all the visualization processes on parallel computers that hold target numerical data and transfers compressed visualized image over a network to a user's terminal. The volumes of data that are transferred over the network and manipulated on the user's terminal are much smaller than in the conventional visualization methods and large volumes of data are efficiently visualized in wide-area network environments.

## 4 Grid Visualization APIs

### 4.1 Overview

Our grid visualization system provides two API levels for the users. One is the Visualization Library API, which is used to provide a server module the visualization capability. Typically, this library is called from user's simulation program (solver) and provides the various visualization functionalities. The second API is the Grid Visualization Services API, which is a wrapper API of Grid Services for visualization. This API is used to access the function of a grid visualization server from a client program.

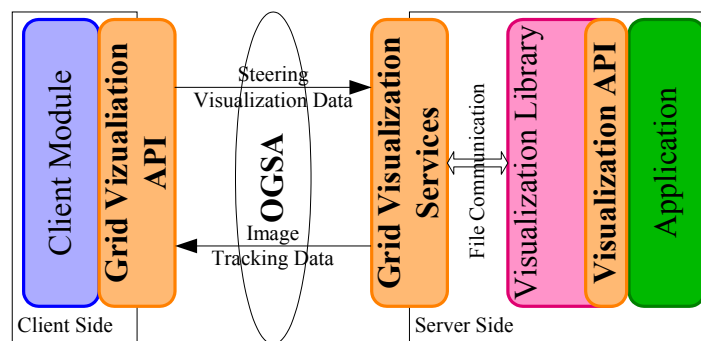


Figure 1: Global overview between client and server side with the Visualization APIs and Grid Services. The client side can send steering and visualization data and the server side can send image and tracking data. The application (solver, reader, etc.) is linked to visualization library with the Visualization API.

## 4.2 Visualization Library API

### 4.2.1 Overview

Visualization Library API is used to make visualization-enabled a server module. Its only and sole goal is to provide a uniform visualization-interface to an application; it is not Grid service provider. The API has a high visualization process level. There are several types of server module, two major examples:

- Visualization module for post-processing. After calculation, this module reads output data from a simulation program and generates visualized images. Visualization Library is incorporated to the program that reads calculation result data.
- Visualization module for concurrent computation. Incorporating the visualization library into a simulation program directly can do this. As this

module is a simulation program with visualization functionality, visualization can be done sequentially with the calculation, in other term real-time.

The API has a front-end with a number of operations enabling the integrating program to visualization (see Table 1), these operations only focus on the content (simulation data).

Operations	Description
GVS_INIT_XXX	Initialization operations to set the different data definitions. In the current model XXX can be replaced with: [SCALAR, REAL]
GVS_INIT	Initialization of the visualization module
GVS_XXX	Set the visualization data during the simulation. The current model supports: [ATOM, QUANTUM, ATOM_SCAL_INT, ATOM_SCAL_REAL]
GVS_MAIN	Visualization control.
GVS_TERM	Visualization termination.

Table 1: General API Operations

We propose a visualization library that integrates the API operations. The current API and library does not provide the back-end programmable rendering parameters (i.e. camera settings, lighting, etc.). At the time being a configuration file controls the rendering parameters such as camera position, lighting, etc. In addition, the API may extend the proposed operations to include a wider range of visualization settings in the future. The current proposition is focused on the molecular science.

#### 4.2.2 Use Case

In the framework of the NAREGI project [14] we have developed a simple molecular viewer for molecular orbital calculation in post-processing mode. This solution uses UNICORE [2] for both the client and server side grid-control.

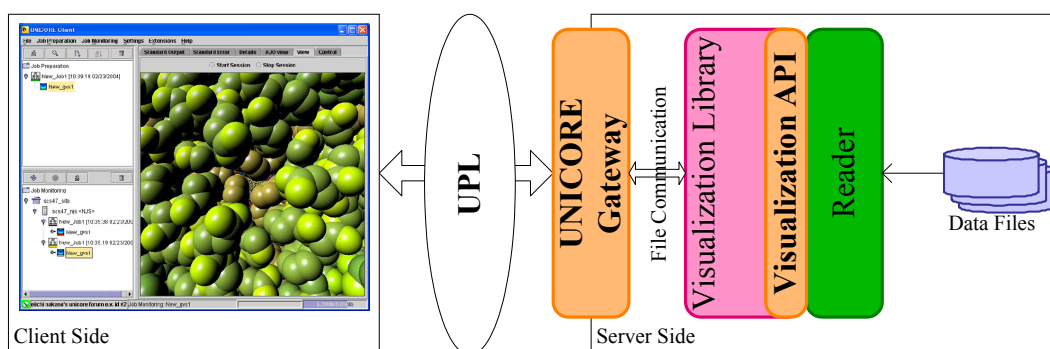


Figure 2: UNICORE molecular viewer is based on UPL (UNICORE Protocol Layer) communication (OGSA not being available yet). The server is launched in batch mode. Communication between the gateway and solver is done with files.

When the integration of the Visualizing Library is done in the reader application, the pseudo code looks like:

```
GVS_INIT_XXX
GVS_INIT
// Solver main loop
GVS_ATOM
```

Library initialization.

Give the visualization library the current simulation

GVS_QUANTUM	data set needed for rendering.
GVS_MAIN	Visualization and image production.
GVS_TERM	Terminate the application.

Example 1: Pseudo code of the server with the API calls. Between the API calls, the normal code specific to of the reader is present.

## 4.3 Grid Visualization Services API

### 4.3.1 Overview

Grid Visualization Service API is provided to access visualization functions via the Grid Services (GS) interface. This API can be used by a client module as well as by any other programs that require visualization capability at a high-level of abstraction over the Grid. The server side services will mirror the different function so that the orders can be transmitted to the underlying program.

In order to cover the different services, the API is divided in generic sub-services:

Service	Description
Concurrent Visualization	Visualization can be done simultaneously with computation.
Post-processing Visualization	Visualization based on the output files of simulation program
Database Visualization	Visualization based on the data stored in databases.

Table 2: Generic Visualization Grid Service

Most services are independent of the simulation used. The implementation focuses on concrete services, like the molecular visualization or coupling simulation visualization, which offers a set of specialized calls.

This API does not intend to provide low-level visualization capability like OpenGL or DirectX. Its goal is to provide high-level semantic for rendering parameters end content. The generic grid services APIs have a number of default operations such as:

Operation	Description
setCamera	Set camera parameters (Eye position, reference point, etc)
setImageSize	Set visualized image size.
setLight	Set a light parameter set.
setColorTable	Set a color table parameter set.
setBackground	Set image background color.
setDataFile	Set visualized data file, any data specific to the simulation.

Table 3: Visualization Grid Service API.

In addition to predefined calls, the APIs must provide a mechanism to pass application specific parameter sets. If the API wraps, for example, a rendering tool such as PyMOL (molecular viewer) it could be possible to order specific rendering commands for that application in particular.

Example for post-processing application, the services can be called in the client program as followed:

```

// Generate an instance of Post visualization service.
PostVisualizationService pvs= new PostVisualizationService();
// Set data file which contents is visualized
pvs.setDataFile("gsiftp://localhost:2811/tmp/file1");
// Setting Camera parameter
pvs.setCamera(...);
// Setting a light
pvs.setLight(...);
// Generate a visualized image
byte[] imageData= pvs.getImage();

```

Example 2: Java example to call a post-processing visualizer in a program.

In this simple java code, post-processing visualization service is called and visualization for the data file 'file1' is performed. Though this simple program contains only simple API calls, more detailed and complicated parameter setting for specific visualization tool like molecular visualization tool can be done.

These APIs proposals are grid-specific for the moment, however if the API is well designed and properly masked behind a design pattern, it could be applied to other services such as pure web services or even a proprietary protocol implemented behind the API.

### 4.3.2 Use Cases

To illustrate the possibilities of the proposal, two possible configurations are proposed. The first case is with a parallel post processing on a remote machine. This use case is well suited when the data files are huge or there are many data files. In this case, by using many computation resources, visualization is done in parallel. This might be useful for parameter survey. Though, in this figure, data files are located on several remote servers, data files can be located on one or more machines or even a local machine.

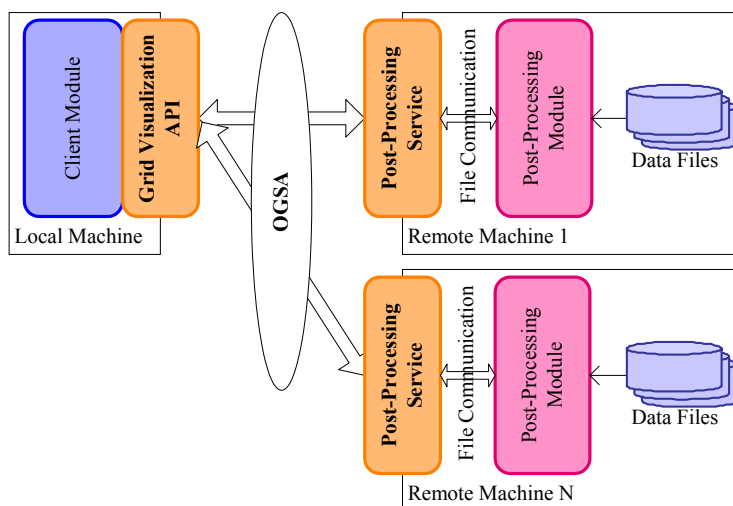


Figure 3: Parallel post-processing on different the remote machines. If the data files must be transferred to the remote machines, a file transfer service must be used; Globus Toolkit proposes such service. The post-processing module can use the Visualization Library.

The second use case concerns parallel concurrent visualization on remote machines. Parallel visualization is used when different solvers runs in parallel but the final visualization is done with one single image.

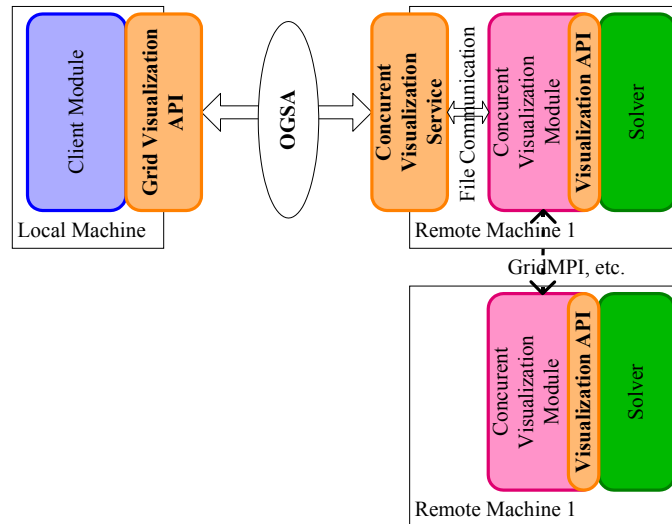


Figure 4: Parallel concurrent visualization on remote machines. The communication between the different server modules can be done with GridMPI or any other local vendor MPI. The visualization module integrates the Visualization API to let the solver call it.

## 5 Discussion and Future Work

The proposal made here is a first step, we are aware that this solution targets a specific range of problems we need to treat. However this project serves as a good base for future extensions. In the future, we would like to realize a standard general-purpose framework for Grid visualization service. Various visualization services provided under a unified framework, for example: Parallelized visualization, incorporation of existing tools into the framework, and visualization of integrated databases.

Various visualization parameters can be categorized into the following three:

- *Universal parameters:*  
Various visualization applications for analysis fields commonly use these types of parameters. View parameters such as camera position and view direction are included in this category.
- *Field-specific parameters:*  
These parameters are commonly used by various visualization applications for specific analysis field. For example, parameters for molecular model display are specific for nano-science and drug design.
- *Software-specific parameters:*  
These parameters are only used by specific visualization software. Parameters for software-specific tools and functions are categorized here.

Our focus is especially on the first two categories (universal parameters and field-specific parameters). In the visualization, these two categories occupy much space and the general-purpose framework for Grid visualization service adopting standard API for these categories has significant meaning.

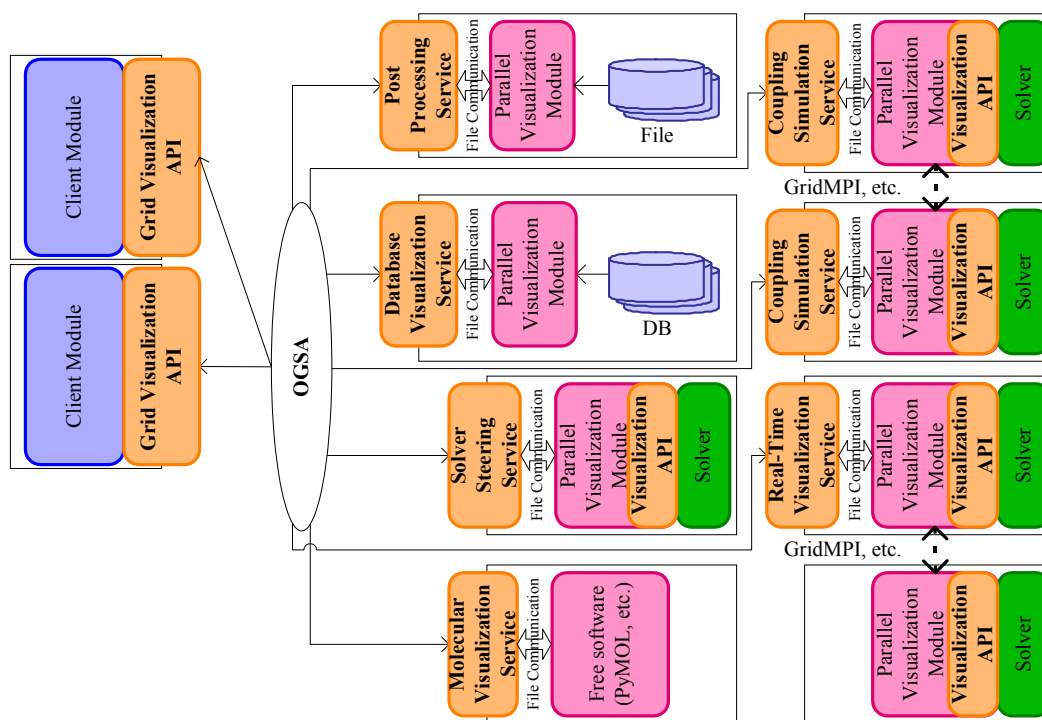


Figure 5: Framework for Grid Visualization Services

More concretely, these service interfaces bring several benefits such as making it easy to integrate existing visualization tools into image-based remote visualization systems, enabling visualization clients to connect with various visualization tools, enabling visualization tools to be used through various types of clients, realizing high interoperability of visualization functionalities in Grid environments. The generalized visualization grids service aims to become a Grid standard.

## 6 Conclusion

We have described a solution that enables Grid and Visualization to join forces. Our proposal enables a client to connect to a simulation server and control the visualization process; in counterpart the server returns an image with all operations done over the grid infrastructure. The connection between the client and server can be done in an easy and independent manner. Finally, the know-how accumulated paves the road for future extensions and generalization of the concepts; if the API proposal is properly crafted and is high-level enough it should not be limited to grid-specific implementation but open the way for other middlewares, the user should be unaware of the middleware used, he cares about the end points of his/her workflow.

## 7 References

- [1] The Globus on the Web: <http://www.globus.org>
- [2] The UNICORE on the Web: <http://www.unicore.org>
- [3] IBM Grid Computing e-business home page: <http://www-1.ibm.com/grid>
- [4] The Reality Grid Project on the Web: <http://www.realitygrid.org>
- [5] The UK e-Science Program on the Web: <http://www.rcuk.ac.uk/escience>
- [6] J. M. Brooke, P. V. Coveney, J. Harting, S. Jha, S. M. Pickles, R. L. Pinning and A. R. Porter. Computational Steering in RealityGrid
- [7] The GEMSS on the Web: <http://www.gemss.de>
- [8] The UniGrids on the Web: <http://www.unigrids.org>
- [9] The VISIT on the Web: <http://www.fz-juelich.de/zam/visit>

- [10] The Forschungszentrum Jülich: <http://www.fz-juelich.de/portal/home>
- [11] Kenichi Miura, The NAREGI Grid Applications Environment.
- [12] T. Takei, J. Bernsdorf, N. Masuda, and T. Takahara, “Lattice Boltzmann Simulation and Its Concurrent Visualization on the SX-6 Supercomputer”, Proc. 7<sup>th</sup> International Conference on High Performance Computing and Grid in Asia Pacific Region, 2004.
- [13] The TeraGrid project on the web: <http://www.teragrid.org>
- [14] The NAREGI home page: <http://www.naregi.org>
- [15] The RVSLIB product: [http://www.sw.nec.co.jp/APSOF/SX/rvslib\\_e](http://www.sw.nec.co.jp/APSOF/SX/rvslib_e)