

Grid-Enabled Computational Fluid Dynamics using FlowGrid*

Jan Wendler, Florian Schintke

Zuse Institute Berlin (ZIB)
{wendler,schintke}@zib.de

Abstract

We present an architecture for Computational Fluid Dynamics (CFD) applications, that we developed for the FlowGrid project. FlowGrid revolutionizes the way CFD simulations are set up, executed and monitored. In this project several Grid centers across Europe develop and validate their software for Grid-based CFD computations. The 'CFD Virtual Organization' of FlowGrid provides industrial end users easy and flexible access to CFD resources.

In this paper we describe the overall FlowGrid architecture and dig into the details along a typical job execution inside FlowGrid. We then outline how a generic CFD Grid service can be set up.

1 Introduction

Computational Grids [3, 6] for CFD simulations, allow the flexible sharing of computational resources across small enterprises to satisfy peak requests with fewer overall licenses by providing on-demand computational power. The downside is that CFD simulations need synchronous communication between subjobs during runtime, which makes it challenging to execute these jobs in Grid environments with potentially high latency links between the different sites.

The FlowGrid consortium consists of four industrial partners, who want to make use of the Grid to compute their CFD simulation, and two technology providers. The industrial partners are Skoda VYZKUM, the Central Research Institute of the SKODA concern; CERTH/CPERI, a leading research organization in Greece; HSVA, Germany's hydrodynamics research and development facility; and the Fluid Mechanics Group at the University of Zaragoza (Spain). The two technology providers participating in the project are Symban Power Systems Ltd, a commercial software company and vendor of the CFD software APUS; and the Computer Science Research department at Zuse Institute Berlin (ZIB).

Some of the applications of our industrial partners are shown in Figure 1 including flow simulation across ships and trains, and flow inside combustion engines. These problems are first partitioned and then simulated on distributed resources across our Grid, using APUS CFD solver from Symban Power System Ltd.

This Grid is composed of several systems that supply computing power, data storage and other resources. Because CFD simulations are computationally intensive, our main

* This work was funded by the EU FlowGrid project IST-2001-38433.

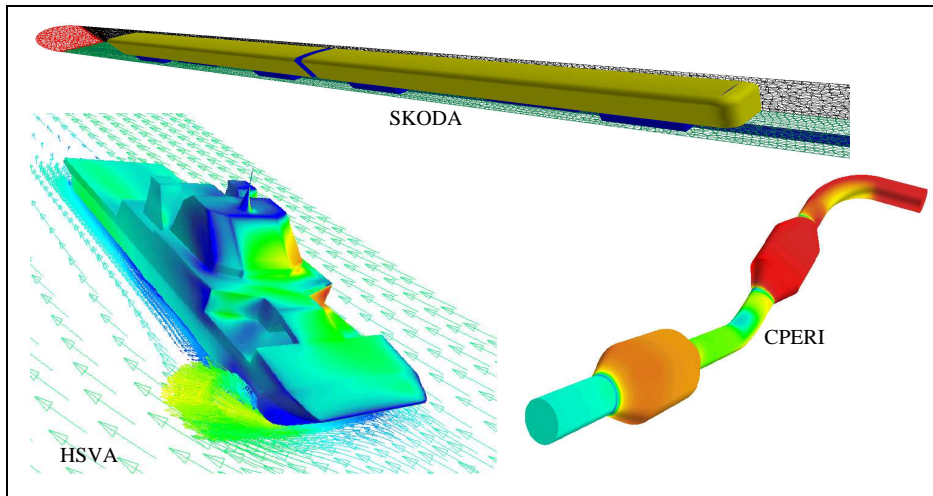


Fig. 1: Some CFD Applications that use FlowGrid.

focus is on the efficient use of computing resources. On several partner sites clusters are installed and provided to the Grid. Each of these clusters is operated by a resource management system (resource manager) such as PBS, LSF or Condor.

2 FlowGrid Architecture

2.1 Components of the Architecture

The components of the FlowGrid architecture are organized into four layers: actor, front-end, middleware and backend. Figure 2 depicts these layers and the distribution of the components across them. The dependencies and interactions between the components are also shown. They are described in more detail in Section 2.2.

The *actor layer* consists of actual persons taking different roles in the interaction with FlowGrid: resource providers who provide their clusters to the Grid, administrators who manage the Grid and subscribers who use the Grid resources. The *front-end layer* consists of two graphical user interfaces. A native Windows interface called GenIUS and a web portal, both developed by Symban Power Systems Ltd. In the *middleware layer* our FlowServe system manages the jobs on behalf of the user. This layer also contains a central database management system (DBMS). At the bottom, the *backend layer* contains the resource managers for the different clusters. On these clusters runs the MPI CFD backend, developed by Symban Power Systems Ltd.

2.2 Interactions between the Components

The dependencies and interactions between the components are depicted in Figure 2. The purpose of each interaction (numbered in the Figure) is shortly described.

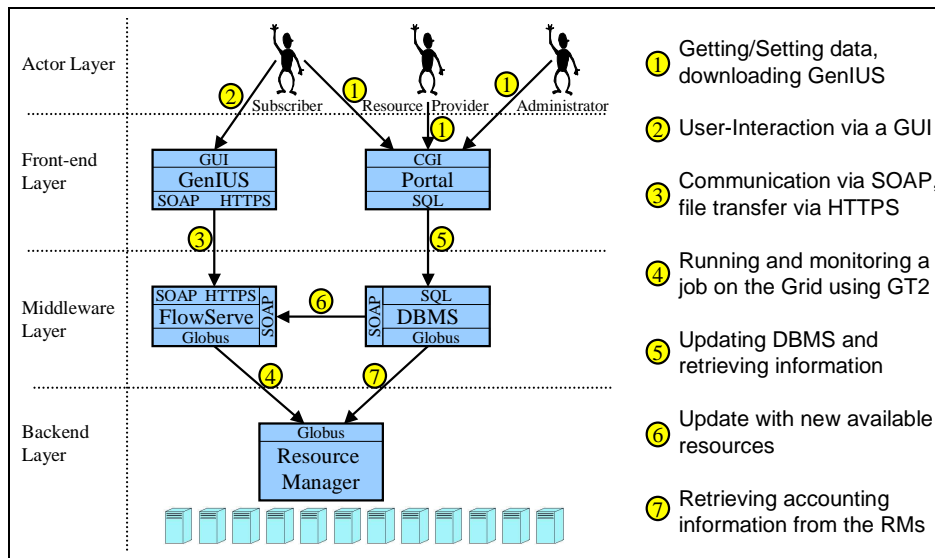


Fig. 2: The FlowGrid architecture is organized in four logical layers.

- ① To offer cluster resources to the Grid resource providers use a web interface. Resource providers can set, read and modify administrative data of the provided clusters. Using the web interface, the FlowGrid Administrator has the ability to add or remove resource providers from the FlowGrid system. Subscribers (end users) can use the portal to download the GenIUS client and to request certificates.
- ② With his certificate the subscriber, i.e. the end user, can use the graphical user interface GenIUS to set up and run CFD simulations on the Grid. Inside GenIUS the user can partition his simulation, generate a credential for the lifetime of the simulation (called *user proxy*) and submit the CFD simulation.
- ③ The job execution order is sent from the user interface to FlowServe using SOAP for control communication and HTTPS to transfer the input and output data. The protocol between GenIUS and FlowServe is described in detail in Section 3.
- ④ FlowServe itself uses Globus [1] to distribute the CFD simulation across the Grid. Furthermore, FlowServe monitors the CFD simulation by running small control jobs. Control jobs are also used to obtain the current status of a cluster including the number of free processors and the number of waiting jobs.
- ⑤ The portal uses SQL to retrieve and update data of the DBMS. The central DBMS stores data on the status of resource providers, clusters, FlowServe components, subscribers and executed jobs.
- ⑥ If new clusters are added to FlowGrid their contact information is announced to each FlowServe using SOAP.
- ⑦ In fixed time intervals a component of the DBMS queries the resource managers to retrieve the jobs they have executed. This information is used for accounting.

2.3 Special Features

Progress Monitoring Using FlowGrid, preliminary results from the Grid application can be retrieved. The Grid application stores its progress information in temporary files. These files are retrieved at fixed time intervals by FlowServe (step ④ of Figure 2). For the retrieval of progress information FlowServe submits small control jobs at the locations where the Grid applications are running. Such control jobs are executed immediately. They are not put into any queue. The control jobs are running on the frontend of the cluster whereas the grid applications run on one or more of the clusters nodes.

Windows-Linux Integration. With the FlowServe system the user can run jobs on the Grid using GenIUS, a native Windows application. We connect the native Windows client program (not just a web portal) with a Grid that usually consists of Unix-compatible computers.

Certificates. For mutual authentication between the components of the FlowGrid system, all actors, FlowServe components, resource managers, and the DBMS own certificates, which are provided by a certificate authority.

Scalability: The FlowGrid system is fully scalable. There is only one central component, the DBMS, which only performs administrative tasks. All other components can be distributed and divided into subcomponents. Usually one FlowServe component is responsible for several GenIUS components, several clusters are used by several FlowServe components and one or more portals interact with the DBMS.

If the administrative tasks of the DBMS become a bottleneck, it is even possible to split the FlowGrid system into several FlowGrid systems, which may then be used by different but possibly overlapping virtual organizations.

2.4 Typical Job Execution inside FlowGrid

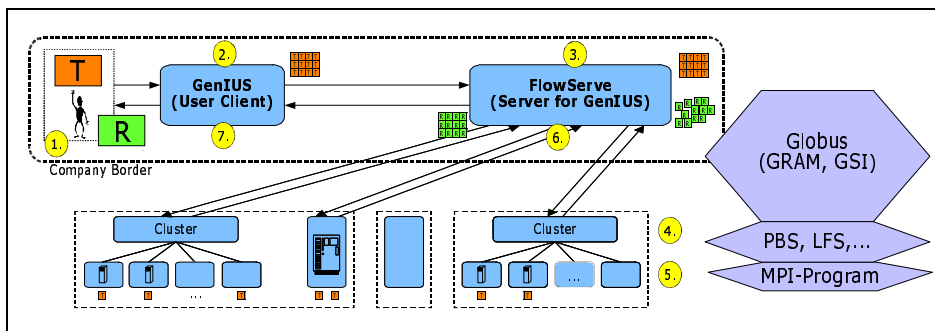


Fig. 3: A Typical Job Execution with FlowGrid.

In order to demonstrate how the FlowGrid system works, Figure 3 outlines a typical, but abbreviated job execution inside FlowGrid. This execution covers the steps ②, ③ and ④ of Figure 2.

- ① The subscriber (end user) has a task T to execute.
- ② The subscriber uses GenIUS to partition the task T into several subjobs T_1, \dots, T_n , by using an external mesh partitioner. He then uses GenIUS to generate a user proxy and to send an execution order to FlowServe.
- ③ FlowServe receives this order and distributes the subjobs T_1, \dots, T_n to appropriate clusters by contacting their resource managers using Globus GRAM and Globus GSI [5].
- ④ The resource managers schedule the subjobs using PBS, LSF or other batch systems.
- ⑤ The subjobs T_1, \dots, T_n start their execution. The subjobs are MPI programs that communicate in fixed time intervals with each other to exchange boundary data.
- ⑥ During the execution of the subjobs, FlowServe retrieves preliminary results from the clusters (not shown in the Figure), which are presented to the user via GenIUS. This allows the user to view the solution process and convergence already during job execution. After all subjobs are finished FlowServe retrieves the results R_1, \dots, R_n .
- ⑦ Finally the results R_1, \dots, R_n are sent back to GenIUS, which generates a final result R that is then presented to the subscriber.

3 FlowServe Protocol

FlowServe executes jobs for the subscriber using the subscriber's timely limited proxy certificate. FlowServe acts as a service so that all interactions are initiated by the subscriber.

One of our goals was to make FlowServe as open and general as possible, so we implemented it as a web service. Unlike most other web services FlowServe not only reacts to requests from subscribers, but also proactively performs several other functions. These functions are triggered repeatedly at fixed time intervals, though they can also be triggered by the subscriber directly.

3.1 Time-triggered Functions

The goal of our time-triggered functions is to speed up the response time of FlowServe to requests by GenIUS. Data from the resource managers are retrieved in advance and cached for a certain amount of time. Furthermore complete reports are generated in advance. These cached data are provided on demand without further latencies. We have implemented time-triggered functions for the retrieval and report generation of

- resource information data,
- job status information,
- preliminary job results and
- job output data.

3.2 Standard Interactions

The most important interactions between GenIUS and FlowServe are represented in Figure 4. Note that only successful interactions are shown.

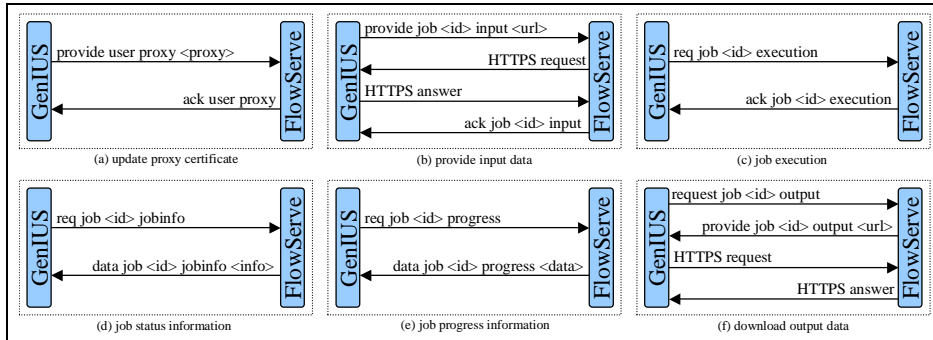


Fig. 4: Selected Standard Interactions between GenIUS and FlowServe.

- Before FlowServe can act on behalf of the subscriber it needs a proxy certificate from the subscriber. The receipt of the proxy certificate is acknowledged by FlowServe (Fig. 4(a)).
- The submission of the job's input data is accomplished in two interaction steps. First the URL of the input data is sent to FlowServe, which then sends an HTTPS request to GenIUS. This HTTPS request is answered with the input data and acknowledged by FlowServe after its receipt (Fig. 4(b)).
- After the proxy certificate and the input data of the subscriber are available, FlowServe accepts the order to execute the job. This order is acknowledged immediately (Fig. 4(c)).
- At any time GenIUS may ask FlowServe about the status of jobs. The response to this request contains the state of the job (no information, sending input, got input data, scheduled, running, receiving output, completed) and if applicable the resource managers where the job is running as well as current time and accounting information from the job (Fig. 4(d)).
- If the job is already running GenIUS can request preliminary results of its execution from FlowServe. The progress data of the subjobs are then composed and transferred back to GenIUS (Fig. 4(e)).
- Finally, once the job is completed, GenIUS needs to retrieve the output data. This is done in two steps. First GenIUS sends a request for the output data, which is answered by a message containing the URL where the output data can be downloaded. Then, GenIUS sends an HTTPS request to initiate the actual data transfer (Fig. 4(f)).

Some other standard interactions, which are not shown in Figure 4 are:

- the acquisition of resource data, which allows subscribers to pre-select clusters for their jobs;

- the deletion of the job’s data on FlowServe;
- the deletion of all the subscriber’s data on FlowServe;
- the modification of the time interval in which FlowServe runs time-triggered functions (intervals can be set individually for every function).

3.3 Extended Interactions

In order to make FlowServe independent from GenIUS and allow FlowServe to be utilized by other CFD applications such as FLUENT, StarCD or CFX or even non-CFD applications, FlowServe supports additional functions which can be used by arbitrary frontend-solver pairs.

With these functions the frontend user interface can specify progress and output files. Furthermore any job specific files can be retrieved from the backend clusters during runtime. We currently discuss how interactive input data can be provided to the backend during the simulation (in case of parameter changes) in general.

4 Generic CFD Grid Services

Recent standardization efforts within the Global Grid Forum (GGF) have produced the Open Grid Service Infrastructure (OGSI) [7, 2]. OGSI defines standardized interfaces for the creation, usage and termination of services and builds on web services. With FlowServe we provide a web service that contains already most of the features of an OGSI-compatible Grid service. We focus on two of these features here: life-time management and service data.

Whenever the user provides his proxy to FlowServe, a personal user service is created whose lifetime is limited to the lifetime of the proxy. The service’s lifetime can be extended by the user with an update of his proxy or terminated by sending a user delete request to FlowServe. In addition to user-specific services, a job-specific service is created by the provision of input data. Job-specific services can be terminated by sending a job deletion or user deletion request to FlowServe.

Within OGSI, service data is used to represent the state data from a service instance. For our job-specific service instances the service data is given by the job’s status information, presented in Fig. 4(d). For the user-specific services the service data may contain accounting information of the last executed user jobs.

5 Related Work on Progress Monitoring and Visualization

One of the less prominent but still important aspects of Grid computing is interactivity, which includes activities such as retrieving progress information from running Grid applications before they are completed. To enable on-line visualization of the actual computation, the Grid application needs to interact smoothly with the visualizer. This means that the application needs to be prepared to interact with the visualizer program either directly or through an intermediate Grid service. With the GVK (Grid Visualization Kernel)[4] the progress information is transferred through an intermediate service.

In order to take advantage of GVK the Grid application has to implement a special interface, which makes it necessary to change the source code of the Grid application. Unfortunately this approach limits the application to programming languages that are already supported by the GVK. Instead of dealing with this limitations, we chose to extract progress information from the Grid application by running small control jobs at the locations where it is running. This free us from dependence on any programming language or major changes to the source code of the Grid application.

6 Conclusion

In FlowGrid we create a Virtual Organization to run CFD applications on the Grid. In contrast to other Grid solutions, FlowGrid supports monitoring of preliminary results during runtime. FlowServe is open and can be used by other CFD systems. FlowGrid also combines the usage of Windows and Linux within a Grid.

Acknowledgments

We thank our partners of the FlowGrid project for the fruitful collaboration. We would especially like to thank Symban Power Systems Ltd for their efforts in the joint design of the FlowGrid architecture, for their GenIUS client, the Solver and the Portal and for providing the first results of simulations run within the FlowGrid project. Also, we would like to thank Guillermo Arias del Rio, who helped us to implement FlowServe, and our research group at ZIB for being open to discussions, support and guidance for our work.

References

1. Ian Foster and Carl Kesselman. Globus: A Metacomputing Infrastructure Toolkit. *International Journal of Supercomputer Applications*, 11(2):115–128, 1997.
2. Ian Foster, Carl Kesselman, Jeffrey M. Nick, and Steven Tuecke. The Physiology of the Grid: An Open Grid Service Architecture for Distributed Systems Integration, 2002.
3. Ian Foster, Carl Kesselman, and Steven Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of Supercomputer Applications*, 15(3), 2001.
4. P. Heinzlreiter and D. Kranzlmüller. Visualization Services on the Grid: The Grid Visualization Kernel. *Parallel Processing Letters*, 13(2):135–148, June 2003.
5. Laura Pearlman, Von Welch, Ian Foster, Carl Kesselman, and Steven Tuecke. A Community Authorization Service for Group Collaboration. In *Proceedings of the IEEE 3rd International Workshop on Policies for Distributed Systems and Networks*, 2001.
6. A. Reinefeld and F. Schintke. Concepts and technologies for a worldwide Grid infrastructure. *Lecture Notes in Computer Science*, 2400:62–71, 2002.
7. Tuecke, Czajkowski, Foster, Frey, Graham, Kesselman, Maquire, Sandholm, Snelling, and Vanderbilt. Open grid services infrastructure (ogsi) v. 1.0. Technical report, Global Grid Forum, 2003.