



Executing and observing CFD applications on the Grid

Jan Wendler*, Florian Schintke

Zuse Institute Berlin (ZIB), Takustraße 7, D-14195 Berlin, Germany

Available online 28 October 2004

Abstract

We present the FlowGrid system, that allows Computational Fluid Dynamics (CFD) simulations to be executed in Grid environments. Using this system, users can observe online the progress of their simulation by looking at intermediate results, that are visualized in the graphical user interface. Several Grid centers across Europe currently use and validate the system with their CFD computations and build a ‘CFD Virtual Organization’ to share their resources and balance their processing load.

We first describe the overall FlowGrid architecture, highlight its special features and present the system along a typical job execution. The Grid infrastructure, i.e. FlowServe, is presented in detail, a description of the accounting system is given and experiences with the FlowGrid testbed are provided. Finally, we provide evidence that the results can be used as a generic CFD Grid service.

© 2004 Elsevier B.V. All rights reserved.

Keywords: Computational fluid dynamics; Grid computing; FlowGrid

1. Introduction

Using the concept of Computational Grids [1,2] for Computational Fluid simulations (CFD) is promising, because it allows the flexible sharing of computational resources across enterprises. To satisfy peak requests, computational power is provided on-demand from cooperating enterprises. Instead of the necessity to own hardware to support peak requirements, only hardware for the average load is required, which often also reduces the number of necessary licenses. The challenge of applying the Grid approach to CFD simulations is

that: (1) CFD applications need many synchronous communications between subjobs, but in Grids we have to handle with potentially high latencies, and (2) CFD users often want to see the actual progress of their simulation interactively, but Grid environments today are dominated by the paradigm of batch processing.

The FlowGrid system was developed by four industrial partners, who make use of the Grid to compute their CFD simulations, and two technology providers. The industrial partners are in particular Skoda VYZKUM, the Central Research Institute of SKODA; 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 are Symbian Power Sys-

* Corresponding author.

E-mail addresses: wendler@zib.de (J. Wendler), schintke@zib.de (F. Schintke).

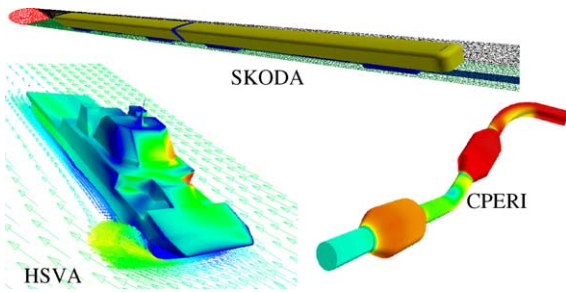


Fig. 1. Some CFD applications that use FlowGrid.

tems Ltd., a software company and vendor of the CFD software APUS; and the Computer Science Research department at Zuse Institute Berlin (ZIB), which provided their Grid expertise and developed the Grid specific parts for FlowGrid.

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

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

main 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) like PBS, LSF or Condor.

2. FlowGrid architecture

2.1. Components of the architecture

The components of the FlowGrid architecture are organized in four layers: actor, front-end, middleware and backend. Fig. 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 APUS-CFD and a web portal, both developed by Symban Power Systems Ltd. In the *middleware layer*,

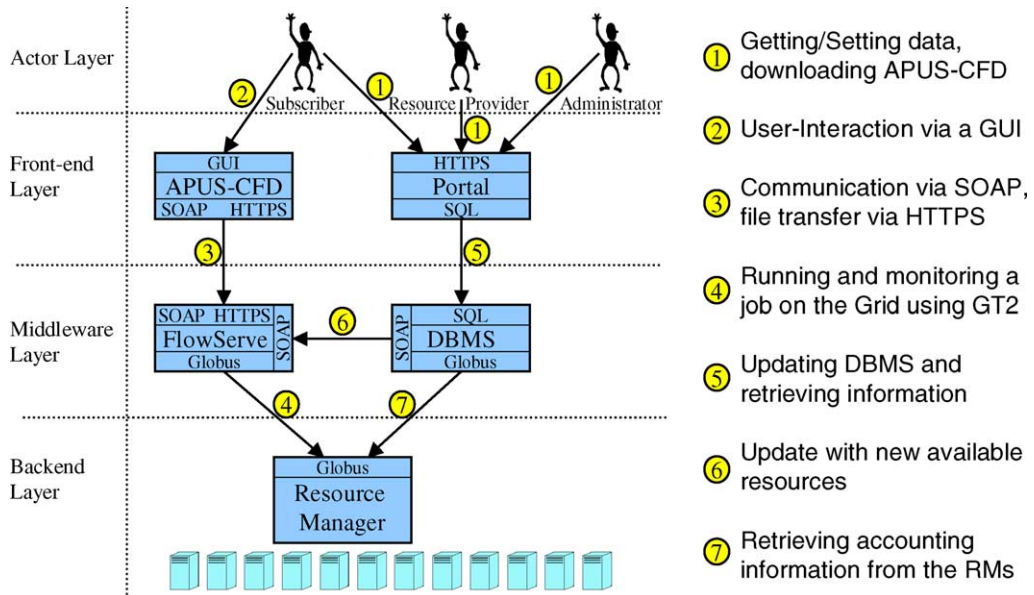


Fig. 2. The FlowGrid architecture is organized in four logical layers.

our FlowServe system manages the jobs on behalf of the user. This layer also contains a database management system (DBMS) per virtual organization using FlowServe. At the bottom, the *backend layer* contains the resource managers for the different clusters. On these clusters runs the MPI CFD backend APUS-CFD.

2.2. Interactions between the components

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

- 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 APUS-CFD client and to request the generation of a certificate. The certificate is automatically saved in a MyProxy database using the username password pair of the user, so that the users do not need to care about it.¹
- ② The end user can use the graphical user interface APUS-CFD to set up and run CFD simulations on the Grid. Inside APUS-CFD, the user can partition his simulation and submit the CFD simulation.
- ③ The job execution order is sent from the user interface to FlowServe using SOAP over HTTPS for control communication and HTTPS directly to transfer the input and output data.
- ④ FlowServe itself uses Globus [3] 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. The accounting system is described in Section 4.

2.3. Special features

2.3.1. Progress monitoring

Using FlowGrid, preliminary results from the Grid application can be retrieved. The CFD backend stores its progress information in temporary files. These files are retrieved at fixed time intervals by FlowServe (step ④ of Fig. 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 and are running on the frontend of the cluster whereas the grid applications run on one or more of the cluster's nodes.

Another approach to progress monitoring is given by the Grid Visualization Kernel (GVK) [4] which allows on-line visualization of the actual computation. To use GVK, the application needs to implement an interface of the GVK. Its usage is therefore limited to applications with supported programming languages.

2.3.2. Job control

The CFD application can be controlled to some extent. It is possible to send small control messages to the CFD backend using FlowServe. FlowServe invokes a control-job to create a file with the name of the control message on the location where the solver reads its input data. The solver so far checks for the existence of the control message 'stop' in fixed time intervals. If it detects a file with the name 'stop', it writes its preliminary results and terminates itself. A wide range of application-specific strings can be used as control messages.²

¹ In an earlier FlowGrid version, the users had full control over their certificates. The industrial end users did not agree with this approach; they wanted to have a convenient access to the Grid using only a username and a password.

² The string can contain any letters and digits and is limited to 255 bytes.

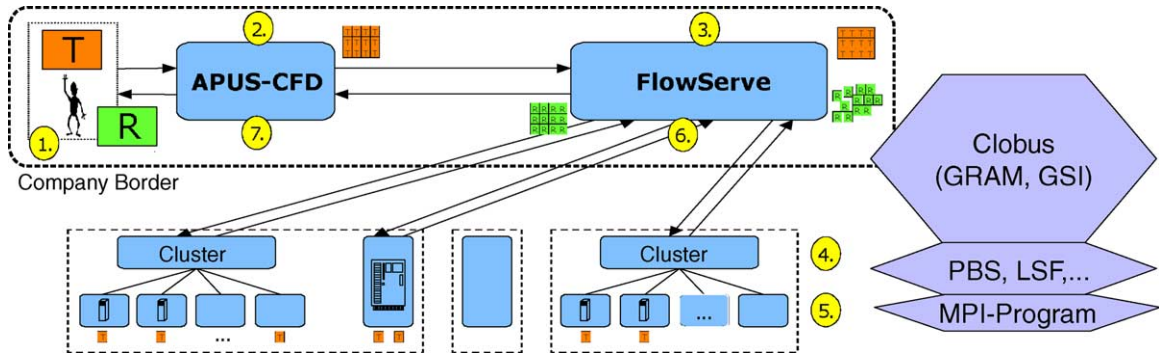


Fig. 3. A typical job execution with FlowGrid.

2.3.3. Interactive jobs

Using the features ‘progress monitoring’ and ‘job control’, jobs can be interactively controlled. The user client has the opportunity to monitor his jobs. In case that the solution of a job does not converge, the user can send a control message to the job, e.g. ‘stop’ to stop the calculation. Then, he can adapt the input data of the job by exchanging some of the input files, for instance, he can adapt the boundary conditions. Thereafter, he can trigger the job to continue using the new values.

2.3.4. Windows–Linux integration

With the FlowServe system, the user can run jobs on the Grid using APUS CFD, 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.

2.3.5. Certificates and authentication

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. The subscribers (end users) authenticate themselves with username–password pairs; their certificates are stored in one or several MyProxy databases. FlowServe retrieves the user certificates automatically from its MyProxy database.

2.3.6. Security

By using SOAP (over HTTPS for control interactions) and HTTPS (for file transfers) as underlying transport protocols for the communication between APUS CFD and FlowServe and between the DBMS

and FlowServe, these communications are secure and encrypted. Between FlowServe and the resource managers and between the DBMS and the resource managers, Globus is used for the communication. The Globus toolkit uses the Grid Security Infrastructure (Globus-GSI) [5] to provide authentication, message protection and encryption using certificates.

2.3.7. Scalability

The FlowGrid system is fully scalable. There is only one central component, the DBMS, which only performs minor administrative tasks. All other components can be distributed and divided into subcomponents. Usually, one FlowServe component is responsible for several APUS CFD 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 FlowGrid job execution

In order to demonstrate how the FlowGrid system works, Fig. 3 outlines a typical FlowGrid job execution. This execution covers the steps ②–④ of Fig. 2.

- ① The subscriber (end user) has a task T to execute.
- ② The subscriber uses APUS-CFD to partition the task T into several subjobs T_1, \dots, T_n . Internally, a common third-party mesh partitioner is used. Then, the user starts the simulation inside APUS-

CFD, which triggers that an execution order is send 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 APUS-CFD. 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 APUS-CFD, which generates a final result R that is then presented to the subscriber.

3. FlowServe architecture and protocol

The components and interfaces of FlowServe are depicted in Fig. 4. We distinguish between internal and external (or *third party*) components.

FlowServe components are depicted as green (dark) boxes including one or more modules (in blue, respectively, light). The round forms represent other Flow-Grid components FlowServe interacts with. Arrows are used to show interactions between modules and components. When an arrow passes through a module, it means that the module acts as a layer, without processing the message itself (for example, the HTTPS layer adds security but does not process a 'create-

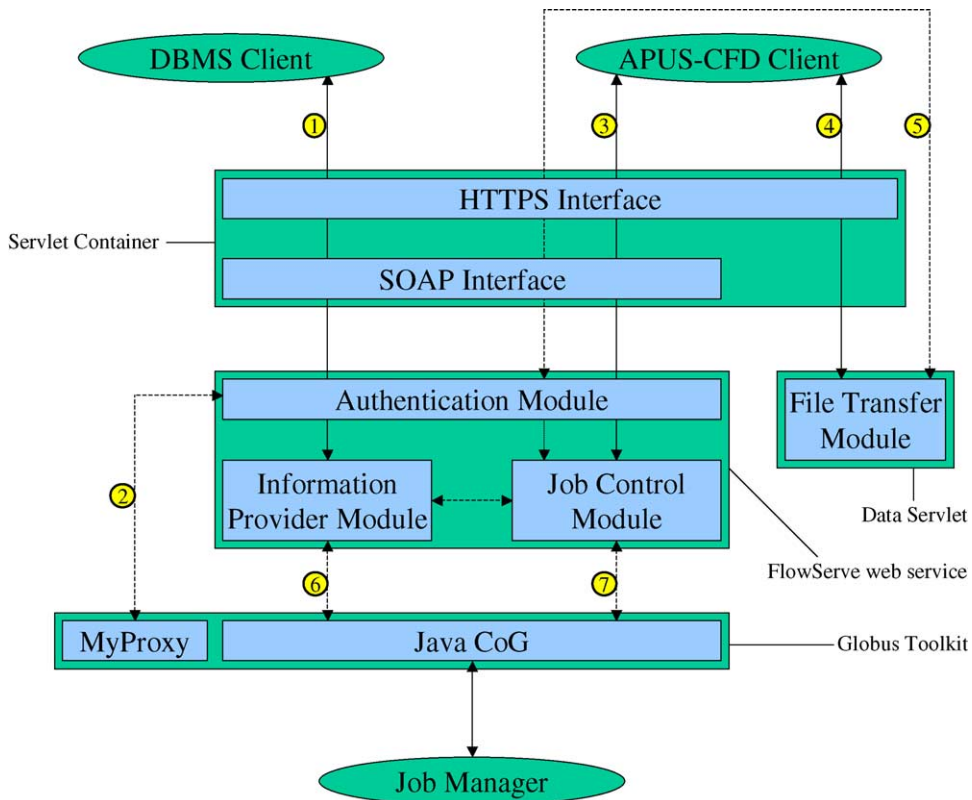


Fig. 4. Components of FlowServe. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of the article.)

Job' request). Dashed arrows represent local interactions.

The interactions of the groups ①, ③–⑤ compose the **FlowServe protocol**. They are initiated by a message from the client (DBMS or APUS-CFD) which is always answered by FlowServe. Almost all interactions require HTTP basic authentication.³ The interactions of the groups ①, ③ and ⑤ are performed using the SOAP protocol over an HTTPS protocol layer. The interactions of group ④ are performed using the HTTPS protocol. All data structures which are exchanged during the interactions are formatted in XML.

Two interactions are implemented between *the DBMS client and the FlowServe web service* (interaction ① of Fig. 4): the interaction 'updateResourceManagerList' is used to inform FlowServe about the static properties of the available FlowGrid resources and with the interaction 'registerUser' the user certificates are registered.

Between *APUS-CFD and the FlowServe web service* (interaction ③ of Fig. 4), a wide range of interactions is handled by the job control module in order to manage user jobs. These interactions include the creation, execution, status monitoring, progress monitoring, controlling and deletion of jobs. Additionally, the information provider handles the interaction 'requestGeneralResources'. This interaction is used by the APUS-CFD client to retrieve information about the available clusters.

The two methods Put and Get, which are specified within HTTP, are implemented as interactions between *APUS-CFD and the Data Servlet* (interaction ④ of Fig. 4). They are configured with a job-specific URL which is given during job creation time, but can also be retrieved later via the status monitoring interaction.

The interactions between *the Data Servlet and the FlowServe web service* (interaction ⑤ of Fig. 4) are used to support the transfer of input and output data. The interaction 'inputStart' is used to request a location where to save the input data of the job; with the interaction 'inputEnd', the Data Servlet informs the FlowServe web service that the input data have completely arrived and the interaction output is used to request the location of the output and/or monitor data that the user wants to retrieve.

³ Exceptions are the interactions 'requestGeneralResources' and 'requestFlowServeInformation'.

The authentication module uses a MyProxy database to store and retrieve user certificates (interaction ② of Fig. 4). The information provider and the job control modules apply the Java CoG library in order to interface with the Globus toolkit (interactions ⑥ and ⑦ of Fig. 4).

4. Accounting and billing system

The accounting and billing system is distributed among all components of the FlowGrid system.

Resource providers specify the accounting information of their clusters via the **portal**. The *cluster specification* is encoded in XML; it consists of a *cluster name*, *static system properties*, *internal system properties* and *cluster usage costs*. The *cluster usage costs* are used for the accounting and billing system. Within this XML structure, the resource providers specify costs for data transfer, data storage and computation time.

The *cluster specifications* are saved in the **DBMS** and communicated to **FlowServe**. Any time the information of one of the clusters has been changed, the whole *cluster specifications* structure is sent to all FlowServe systems. The time a new FlowServe system is added via the portal, the *cluster specifications* are submitted to the new FlowServe system. The *names*, *static system properties* and *cluster usage costs* are forwarded to **APUS-CFD** via the interaction 'requestGeneralResources' so that the subscribers can base their decision which cluster to use also on cost information.

During the job execution, the **resource manager** on the cluster generates a log file. Among other things, the log file contains information about the data storage size and time as well as information about the computation time and the number of processors used. The information in the log file is used to generate cost information for the user which is transmitted to the **subscriber** via the status monitoring interaction of the **FlowServe** web service. The **DBMS** executes control jobs to retrieve invoice information from the clusters using the log files of the resource manager. The invoice information can be requested via the **portal** by the **resource providers**.

5. Experiences with the FlowGrid testbed

The FlowGrid testbed currently consists of three backend sites, each contributing one cluster for the

computations. All together, the backend computers consist of 104 CPUs. The backend sites are geographically and organizationally dispersed, one cluster belonging to UZ in Spain, the other one belonging to CPERI in Greece and the last one belonging to ZIB in Germany. None of these clusters is dedicated only to the FlowGrid project; they are also used for other tasks. Three FlowServe components are currently installed within the FlowGrid testbed.

The FlowGrid testbed has been used by the FlowGrid partners for the evaluation of the CFD code. The current logging mechanism which supports the accounting and billing system has been only introduced recently. Before that, a preliminary logging system had been used on one of the clusters. During this time period, 87 jobs had been successfully computed on this cluster. Most of the jobs used four to eight processors and took between 5 and 20 min to compute. However, also long jobs which took more than 20 h time had been executed.

6. Generality of FlowServe

To demonstrate the generality of FlowServe, we adapted it to another CFD application from our project partner from Zaragoza. While it took us substantial time to make this code compatible with MPICH-G2, the adaptation of FlowServe to this code took us only 2 h including testing. Based on the strict partitioning of FlowServe into web service, Data Servlet and job control scripts, it was only necessary to exchange the scripts.

The adaptation of FlowServe does not include an adjustment or generation of a graphical user interface like APUS-CFD. Instead, we have used our Testclient, who was developed together with FlowServe, to verify the correct adaptation of FlowServe. Nevertheless, the Testclient is a fully functional client who supports all features discussed in Section 2.3 and can be used together with graphical interfaces. In order to use a full integrated and user-friendly graphical user interface with FlowServe, it has to implement the FlowServe protocol to the client which was outlined in Section 3 (paragraph 5) and is

described in detail in a public project deliverable [6].

7. 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. Especially we thank Symban Power Systems Ltd. for their efforts in the joint design of the FlowGrid architecture, for their APUS-CFD client, the Solver and the Portal. 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 for discussions, support and guidance. This work was funded by the European Commission under grant IST-2001-38433 (FlowGrid Project).

References

- [1] I. Foster, C. Kesselman, S. Tuecke, The anatomy of the grid: enabling scalable virtual organizations, *Int. J. Supercomput. Appl.* 15 (3) (2001).
- [2] A. Reinefeld, F. Schintke, Concepts and technologies for a worldwide grid infrastructure, *Lecture Notes Comput. Sci.* 2400 (2002) 62–71.
- [3] I. Foster, C. Kesselman, Globus: a metacomputing infrastructure toolkit, *Int. J. Supercomput. Appl.* 11 (2) (1997) 115–128.
- [4] P. Heinzlreiter, D. Kranzlmüller, Visualization services on the grid: the grid visualization kernel, *Parallel Process. Lett.* 13 (2) (2003) 135–148.
- [5] L. Pearlman, V. Welch, I. Foster, C. Kesselman, S. Tuecke, A Community Authorization Service for Group Collaboration, in: *Proceedings of the IEEE Third International Workshop on Policies for Distributed Systems and Networks*, 2001.
- [6] J. Wendler, D3.7 Report on the Generalization of the FlowServe Architecture, 2004, <http://www.unizar.es/flowgrid/>.



Jan Wendler graduated and received his PhD in Informatics from the Humboldt-Universität zu Berlin. He currently works as a research staff member in the Computer Science Research Department at the Zuse Institute Berlin (ZIB) and is there responsible for the management of the EU project FlowGrid. Dr. Wendler has performed research in the areas agent-oriented techniques, multi agent systems, case-based reasoning and distributed systems. Currently,

he is interested in the research areas distributed computing and autonomic computing.



Florian Schintke is a PhD candidate with the Zuse Institute Berlin (ZIB). He graduated in 2000 with distinction from the Technical University, Berlin. Since then, he works as a research staff member in the Computer Science Research Department at ZIB and participates in the EU projects DataGrid, GridLab, FlowGrid, GridCoord and CoreGrid. He is a member of the German Computer Science Society and the IEEE Task Force on Cluster Computing. His

research interests are in the areas of distributed data management, scalable grid systems and autonomic computing.