Int J Interact Des Manuf (2015) 9:235–245
DOI 10.1007/s12008-014-0236-1

ORIGINAL PAPER

Computational steering of CFD simulations using a grid computing environment

Manuel García · Juan Duque · Pierre Boulanger · Pablo Figueroa

Received: 20 June 2014 / Accepted: 26 June 2014 / Published online: 18 July 2014

Abstract Simulation of complex phenomena is usually a long computing process and it has been traditionally performed in batch mode on large high performance computing (HPC) systems. However, advances in computer processing and networking capabilities can now be used to monitor and alter simulation parameters whilst it is running. This process is called computational steering. By combining this capability with advanced communication tools, it is now possible for a group of scientists located across the world to work collaboratively while visualising on-going simulations. This raise the possibility that researches can now share their experience and promote new ideas and solutions by exploring collaboratively the solution space of a complex simulation. In this paper, a collaborative computational steering environment specialised to solve CFD problems is presented.

Keywords Computational steering · CFD · Collaborative environment · High performance computing · Virtual wind tunnel

M. Garcia (✉)
Departamento de Ingeniería Mecánica, Universidad EAFIT, Medellin, Colombia
e-mail: mgarcia@eafit.edu.co

J. Duque
Mecanica Aplicada, Universidad EAFIT, Medellin, Colombia
e-mail: jduquelo@eafit.edu.co

P. Boulanger
Department of Computing Science, University of Alberta, Edmonton, Canada
e-mail: pierreb@ualberta.ca

P. Figueroa
Departamento de Ingeniería de Sistemas y Computacíon, Universidad de Los Andes, Bogota, Colombia
e-mail: pfiguero@uniandes.edu.co

1 Introduction

Computational steering has been an active area of research since the 1980s. A survey of early developments in the area can be found in [20]. It presents a comparation of scope, architecture, and user interface of the systems reviewed. A general computational steering environment that uses high performance computing is presented in [19]. A framework for structural design and optimisation can be found in [10], where the fixed grid method is used to speed up the simulation.

Applications to the study of fluid flow conditions around virtual prototypes is not new for the engineering and academic communities. Early developments include the NASA's Virtual Wind tunnel at AMES Research Center [6] which provides a virtual reality environment for postprocessing CFD analysis, and the Java Virtual wind Tunnel developed by David OH at MIT [21]. Kreylos et al. [17] at UC Davis developed a system for visualization and manipulation of ongoing CFD simulations. In their system the user can manipulate visualization primitives such as isolines, streamlines, selectively refine domain regions, and remesh in real time. The system is designed for unsteady two and three-dimensional incompressible flow.

The gViz is a UK e-Science project for visualization and computational steering on the grid [5]. It is based on the traditional dataflow model used by tools like IRIS explorer [9] and VTK [23], and originally proposed by [14]. Besides data flow turned into visualization the gViz model also take care of physical resources from a Grid environment and adds collaboration. Computational steering is treated as a one type of visualization in the gViz platform.

Rank and its group at TU Munich have developed a collaborative computational steering platform based on a distributed architecture composed of a central collaboration

server, an arbitrary number of simulation servers and an arbitrary number of clients [29]. Their system uses the lattice-Boltzmann method as a simulation engine and has been tested in several model scenarios including a collaborative layout of a Heating Ventilation Air-Conditioning (HVAC) system [4], and an indoor airflow simulation for a complex surgery room [22].

Bordegoni et al. [3] developed an interactive system for real time modification of the structure shape via a haptic system. Their main point is to demonstrate that a smooth and effective integration of modeling tools based on haptic interfaces, fluid-dynamics analysis tools, and Virtual Reality visualization systems is feasible in real-time through the use of a proper data model exchange. Finally, Bazilevs et al. [1] proposed a control strategy for Fluid Structure Interaction (FSI) problems based on the information provided by the solution of the adjoint FSI problem. Such control strategies are then used for computational steering in which the computational model is adjusted to include the information coming from the measurement data then the physical system is driven toward a desired behavior.

The designer need to explore rapidly the property of new designs is critical for a world where time-to-market is getting shorter and shorter. This paper presents an overview of a new computing architecture where an Interactive Computational Fluid Dynamics (ICFD) environment is used to optimize collaboratively the airflow around virtual prototypes. The main idea is to constrain the design domain at the early stages of a product development, where highly accurate results are not as important as the fast estimation of the influence of state variables on the design. This computational steering system uses an automated server responsible for managing open-source CFD modules from which the flow simulation is computed and where it is possible to update the simulation parameters on the fly. Data and user instructions are collected by a client applications that connects to the server (located on the same computer or on a High Performance Computing HPC facility) solutions where they can be visualized over time as information arrives. The user can then decide to abort a particular simulation or restart a new one with other parameters. He can also allow the system to continue the simulation until it is completed. Using a powerful HPC computer as a central server, a group of designers can explore rapidly the various solution of a prototype design in parallel. In Sect. 2, a review of the interactive needs and requirements for CFD are presented. Section 3 describes the proposed computational steering architecture and Sect. 4 its implementation for a simple CFD experiment. We will then conclude by describing the pros and cons of the system and how the system can be improved.

2 Interactive CFD needs and requirements

The main idea behind most of the *CFD* simulations is to obtain a global and accurate model of the behaviour of a fluid flow in contact with other fluids or with solids. These flow are in most cases non-steady. The use of non-steady flow simulations is one of the best ways to obtain an idea on the behaviour of a specific virtual design.

ICFD is useful for understanding the physics of the systems and how it responds to variations of the design variables, to optimise the design in terms of energy efficiency or to improve the aerodynamic performance. For *ICFD*, the simulation parameters might be modified according to different classes:

1. *Geometrical*: This parameters are mainly geometric and include:
   (a) *Relative positioning of a body;*
   (b) *Relative Orientation;*
   (c) *Size (Scaling);*
   (d) *Shape modification;*
   (e) *Addition-Subtraction elements;*
2. *Physical*: The nature of these variations is related to the modifications on the physical parameters or phenomenon. The possibilities are:
   (a) *Fluid properties;*
   (b) *Boundary conditions;*
   (c) *Simulation timesteps.*
3. *State of the Simulation*: Setting Start-Stop flags is necessary to control the continuity of the process.

In ICFD all variation parameters are controllable by the user. Most of the geometry changes require a recalculation of the mesh used for the computations and may also introduce other parameter variations such as changing the boundary conditions of the domain. The possible effects of performing parameter changes in the simulation are portrayed in Table 1, as well as the possible chain of events these changes might cause.

A single parameter variation may induce a chain of events that may increase the computational cost, e.g. If a body is added to the model, the whole mesh has to be recalculated, new boundary conditions has to be set over the body (no-slip wall conditions for example). Also, possible changes on the flow regime might be induced and needed to be calculated, etc.

The level of control the user might require to view the solutions is only actualized at the post-processing stage, on the other hand *ICFD* require a level of control and parameter changes that need to be performed during calculation. This

Table 1 Variations on the *CFD* scenario and their respective effects, the first four are geometric changes, the next three are changes to the physical parameters and the last is related with the state of the simulation

Change	Direct effects	Possible induced changes
Positioning	Mesh-recalculation	Boundary conditions, Timestep
Orientation	Mesh-recalculation	Boundary conditions, Timestep
Size	Mesh-recalculation	Boundary conditions, Timestep
Shape modification	Mesh-recalculation	Boundary conditions, Timestep
Addition–subtraction	Mesh-recalculation, Boundary conditions	Timestep
Fluid properties		Timestep, Mesh-recalculation, Fluid flow regime change
Boundary conditions		Timestep, Mesh-recalculation, Fluid flow regime change
Timestep		Mesh-recalculation
Continue the simulation?	Start–stop	

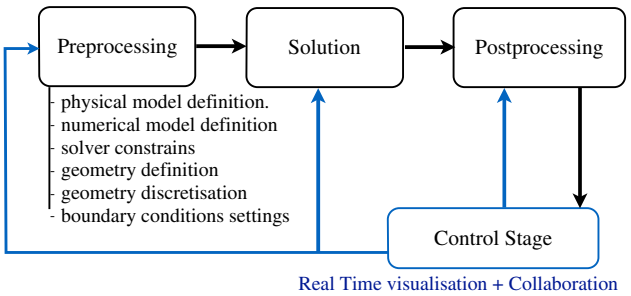


Fig. 1 Flow diagram of a *ICFD* system. The *blue lines* show the feedback by the control stage (color figure online)

new level of control must guarantee simulation stability and the adaptation of the simulation parameters to the current conditions. The flow diagram for an *ICFD* system is shown on Fig. 1. It consists of four stages:

1. *Pre-processing stage* Geometry discretization, physical model definition, and boundary conditions definition;
2. *Solution stage* Given a physical model and a suitable CFD discrete mesh, a particular solver is selected and invoked;
3. *Post-processing stage* As the solver starts producing all the results expected from the solution stage, all data is interpreted and displayed to the user using a standard graphic environment;
4. *Control stage* Parallel to the post-processing stage, the control stage is performed from the results shown during the post-processing stage and is refined by the user if any change during the current *CFD* session needs to be done. If a change to the *ICFD* is requested, the control structures or the user itself must perform a feedback process to all the other stages (*Pre-process, Solution and Post-process*) of the *CFD* simulation, providing the necessary data to continue the simulation.

The possibility to perform any of the system changes shown in Table 1 suggest that the list of changes that arise from an *ICFD* might be grouped into 2 sets:

1. *Non Interactive CFD changes* This set of changes are mainly relating to the initial setup of the simulation;
   (a) *Pre-processing*. Geometry definition setup, discretization information, boundary conditions, physical model, numerical methods, and selection of a solver for these constrains.
   (b) *Post-processing.*
      – *Selected data for displaying* given the massive amount of information the user can access after solving an simulation, choosing which sets of data are to be the best is necessary.
      – *Viewpoint and Scales.*
2. *ICFD needs set.* This set of changes arises from the condition of being interactive during the post-processing via the control stage. This set of needs go hand by hand with the possibilities of variations over the *CFD* scenario described in Table 1.
   (a) *Geometric redefinition:* This subset of changes is mainly relating to the topology and geometry changes of the prototype.
      i. *Body Re-positioning control*: To have control over the relative position between coordinate systems attached to each models involved during the simulation;
      ii. *Body Re-orientation control*: To have control over the relative orientation between coordinate systems attached to each models involved during the simulation;
      iii. *Body Re-sizing control*: To have control over the possible scalability of each models involved during the simulation;
      iv. *Body addition-subtraction*: To have control over the possible addition or subtraction of new/old geometry of the model, which will be positioned, oriented and scaled via the controls described in (i),(ii) and (iii).

(b) *Physical variables redefinition:* This subset of changes is mainly relating to the physical constrains in a *CFD* simulation.
   i. *Fluid properties control*: To have control over the physical properties that define the fluid (viscosity,specific weight, heat, and etc...);
   ii. *Time-step control*: To have control over the time-steps used to solve the simulation;
   iii. *Boundary conditions control*: To have the ability to modify the boundary conditions.

(c) *Change the current state of the simulation*: To have control over the ability of stopping or continuing the simulation.

2.1 CFD and collaborative workspace

Collaborative workspaces are virtual or real in nature depending on how different people work in a collaborative fashion to achieve a common goal. Computer supported collaborative workspaces (CSCCW) are increasingly being used to enhance collaboration between people located at different locations in the world.

The use of common tools such as 3D model navigation and handling is a natural choice and even though the integration of video communication systems between clients is a possibility to achieve a more direct and personal interaction, the imminent appearance of gaps at conceptual expression and transmission of the generated knowledge and ideas is still difficult to avoid

Different schemas to achieve highly efficient collaborative virtual workspaces have been proposed by Billinghurst and Kato [2] and others. A common denominator is that the communication pace can be used as means to minimise the impacts of changing suddenly in a short matter of time the conditions of the workspace. During the present research, a synchronous interaction scheme will be used as default, given the relative ease to control that the variations on the CFD scenario are performed in a sequential order.

3 Computational steering architecture

3.1 Computational steering scheme

The goal of the environment is to develop interaction techniques for users on remote locations using a high speed network. As the users have different specifications of their available computational resources (operative system, computer capabilities, etc.) a Client/Server schema where only some specific data transactions and messages allowed becomes the natural choice for development. Figure 2 shows the basic architecture of the system. The main role of the server is to guarantee a smooth interaction between the users and the solver. The users are located at a local client machine and the server and solver are both located on a remote machines. Figure 3 presents the abstraction of the processes required to guarantee an stable and smooth interaction between the user, the server (simulation/user manager) and solver applications. Three different process involved on the steering environment are identified:

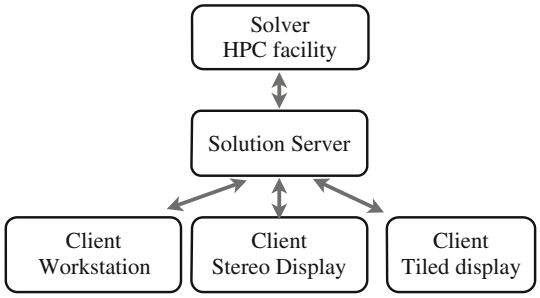


Fig. 2 Global view of the architecture for the CFD computational steering system

1. Data I/O and Steering: This process is handled by the Client. The Client should be able to handle all the defined user data requests and inputs, guaranteeing a fluent interaction and allowing the user to perform the steering of the scenario he/she desires either defining from scratch a new scenario or using a previously defined one.
2. Data distribution and Simulation triggering: This process is handled by the Server (ruby script) that is able to handle the user requests performed from remote locations. The Server is multi-user oriented and the interactions presented on this diagram are only performed with a single user (analogically, the same tasks are performed for the interaction with any user).
3. CFD scenario solving: After the developments of the Solver this is has become a straightforward process when all the data is properly defined and set. The preprocessor/solver are programmed on C++ and can be used as an independent tool when required

It is important to note that to start a new simulation the user has to provide all the geometry files and scenario setup (entry point number 1, Fig. 3). On the other hand, another user might be interested on steer/watch the results of an already defined simulation (entry point number 2 Fig. 3). Finally, entry point number 3 refers to the need of steering an already running simulation previously defined by another user. The number of Client/Server communications and its size were minimised in order to keep network data transfer as low as possible, trying to avoid as many latency and bandwidth related problems. The developed architecture allows several users to steer/access the same datasets from their own clients,

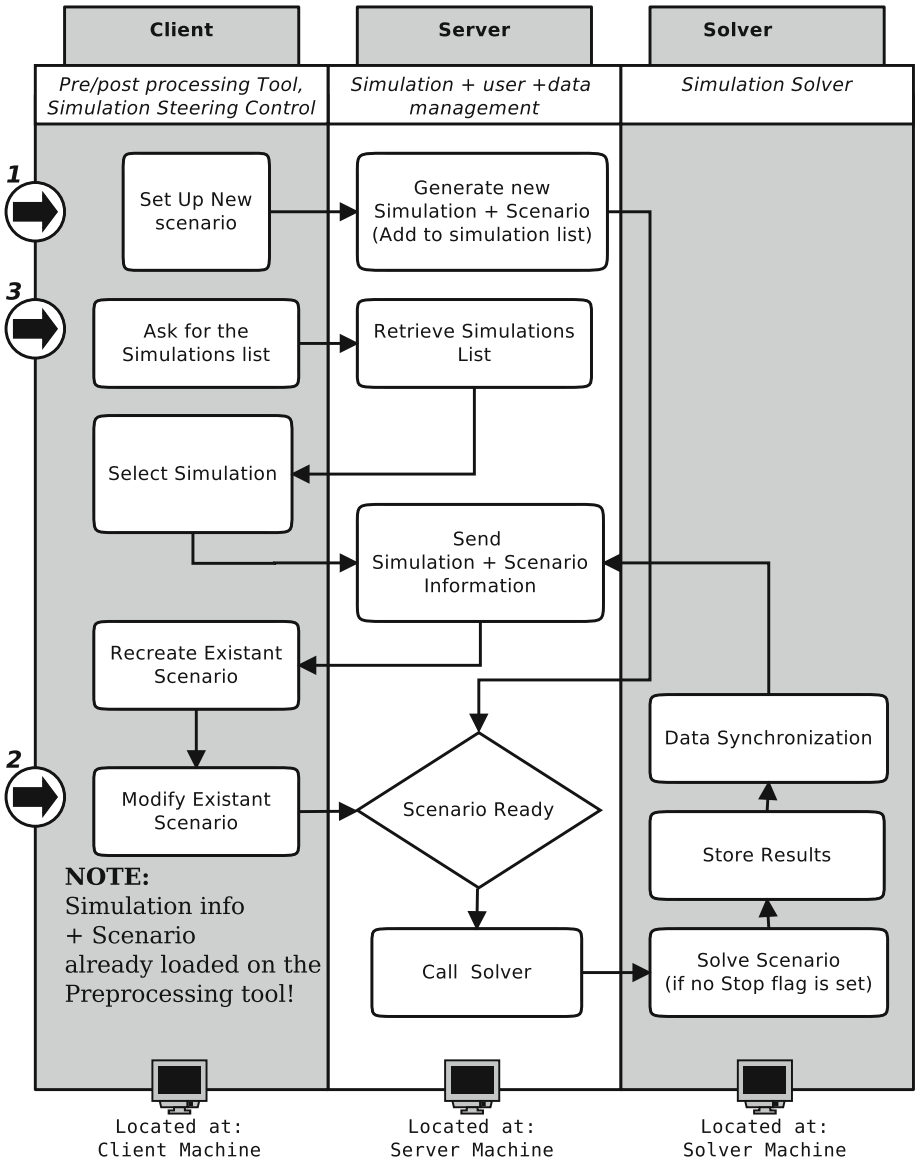


Fig. 3 CFD simulation steering dataflow

providing the basic setup for collaborative design discussions in between the clients via communication services.

3.2 Setting up of a new scenario (virtual wind tunnel example)

Among the different types of fluid simulations, the external flow analysis is one of the most popular due to its application to aerodynamics and hydrodynamics. The main goal of these studies is related with finding the forces acting of solid structures, pressure and velocity distributions around an object, and computing the lift and drag coefficients.

These type of analysis can be made on wind tunnels, but in these days CFD analysis replace much of the experimental part, and experiments are only used to validate simulations. A wind tunnel is basically a closed hall with an inlet and outlet which allow a transversal flow of air across it. They are used in academic and industrial research to test the aerodynamic variables on a particular design case. The idea behind this computational implementation is to be able to steer the different variables and get immediate feedback from the simulation. The steering variables include inlet velocity, boundary conditions, fluid properties, solid body shape, position and orientation.

A transient-state/incompressible/newtonian flow will be used as pilot. The aim of the following sections is to analyse the data requirements to define a generic instance of a Virtual Wind Tunnel (VWT) Scenario, and to estimate the nature of the messages that should be passed from client to server and vice versa.

3.2.1 Fluid domain and bodies

To define a new VWT scenario, it is necessary to establish a proper boundary for the wind tunnel itself, the bodies whose aerodynamic features will be tested, the wind speed at the inlet and other information.

The fluid in a VWT scenario is represented by a hexahedral shaped (box shaped) domain and its orientation should be the same of the cartesian axis of the world coordinate systems. The size of the domain is given by its `length` × `width` × `height`, and its location is given by the location of its geometrical `Center`. They are required to properly define the domain.

Once the VWT domain boundaries are set, the user can start adding *Bodies* to the scenario. A *Body* is a solid object and it is described by its boundary representation (B-REP). Different approaches can be used to define a B-REP, being piecewise planar representations the most commonly used. The location of the `geometric center` or any `fixed reference point` in terms of a triplet of numbers is required as well to establish a reference frame for the body position (Fig. 4).

After the body is located inside the boundaries of the tunnel, the a set of geometric transformations should be applied over it to achieve a correct orientation. The selected approach is to apply a set of `Translations` and `Rotations` of the body respect to the world coordinate system reference frame, using the center of gravity of the body as reference point for the rotation operations.

For some cases, the study case might require to analyse the interaction between various *Bodies* inside the tunnel. Following the procedure presented beforehand, the same operations should be performed to add more *Bodies* to the scenario as presented on Fig. 4.

The geometric information to establish any CFD scenario should be available initially on the client side and should be transferred and stored on the server side, for simulation, reconstruction and distribution purposes among the clients and the solver.

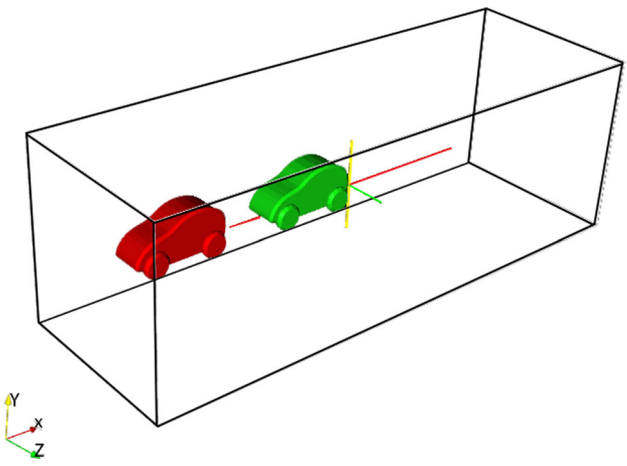


Fig. 4 VWT scenario with 2 bodies

3.2.2 Initial and boundary conditions settings

For the present VWT scenario a basic set of boundary conditions that reflect the problem conditions should be set and parameterised.

For all the tunnel walls and all the faces belonging to the body, a no-slip wall condition (zero velocity) will be set by default. A pressure outlet condition will be used to set the reference pressure at the outlet of the system (atmospheric pressure) and for the inlet of the wind tunnel, a velocity inlet will be used and its magnitude is required as an input parameter, named U_0.

Zero velocity and the same reference pressure it is used inside the whole domain as initial conditions. This implies that the results from the initial timesteps will not reflect accurately the fluid flow, but as time develops, the solution will be more and more accurate.

3.3 Preprocessing

After all the conditions and parameters that define the scenario are defined by the user, the data should be transmitted, gathered and validated inside the solution server to reconstruct the scenario and perform the required domain discretization. Geometry, boundary conditions, initial conditions and physical parameters are adequately integrated by the server and passed to the solution algorithm. The Finite Volume Method (FVM) [7] relies on a mesh to represent the physical domain and hence, a fast and reliable meshing algorithm is required.

In order to speed up the meshing process a Fixed Grid (FG) volumetric mesher was used to discretise the 3D domain. FG meshing algorithms are used as an efficient link between CAD systems and Finite Element/Finite Volume solvers given their low computational cost and versatility [12]. The current implementation uses an optimised parallel FG mesher called Paravoxel. Details of its implementation can be found in [11]. It is important to notice that Paravoxel uses a Convex Hull approach to approximate the geometry of the elements near the boundary of the object, leading to a defeaturing of the model, which could be a problem when accurate calculations are required. However, for early design stages where performing fast-interactive testing to shorten the design domain, this type of Fixed Grid approximation is not only suitable, but an advantage.

The balance between the size of the Fixed Grid and the smallest feature of the solid body will be determinant to guarantee a proper representation of the boundary with the convex-hull elements. An analogy to the Nyquist-Shannon

Theorem for analog signal sampling [24,25] is evident during this process: *to achieve a close representation of the geometry by a fixed grid method, the largest size of an element in the grid must be at least half the size of the smallest feature of the solid body*.

3.4 Solving

Following the preprocessing stage, the data is sent from the client to the Server and from the Server to the Solver. All data should be coupled in terms of the data structures the solver requires for performing the simulation. This step should be transparent to the final user to guarantee an streamlined interactive experience.

For the presented architecture with a VWT scenario, a FVM solver for incompressible steady state newtonian flow is integrated with the preprocessor. The OpenFOAM toolkit [16,28], was used to create a customised solver that was able to handle automatic remeshing with Paravoxel. Also the solver program was designed to listen for any event and adapt the simulation to a client request. Notice, that the basic architecture presented in the previous sections is method independent, hence another solver which is suitable to solve the same physical phenomenon is adaptable, by means of changing the data structures passed on to the solver.

3.5 Postprocessing

During this stage the solution data is sent to the client machines either as a complete dataset or as a simplified video stream. The user must be able to control the viewpoint, colormaps and the amount and nature of the information produced during the solution of the given scenario. On the present implementation the following criteria were taken as reference:

- Basic Requirements: A set of basic filters such as glyphs, stream tracers, contours, isosurfaces, colormaps and plots should be available for the user.
- Environments: A 3D environment where the user is able to travel around the scenario should be available for users with high end computers/large bandwidth capabilities. Given the heterogeneous nature of the clients, a basic video stream using precomputed scenario viewpoints should be available for users with reduced bandwidth or computational resources.
- Tools: Interaction means such as haptic, 3D or multiple axis interaction means should be available for the user to get the most out of the interactive experience.

4 Computational steering implementation

The present section describes an implementation of the previously presented architecture for steering of CFD simulations. The first parts presents the technical requirements and issues faced during the implementation on the server, and clients, and a description of the execution of the system. The final part presents a discussion around the user experiences while using the software.

4.1 Development requirements for the server

In agreement with Sect. 3, the following initial list of requirements for the Server was harvested, summarising the perception and ideas that will be required for this development. In front of each development requirement, the solution proposed to suffice this need will be presented.

- *Stability*: an Unix based operative system was chosen to be the host of the Server service. The stability of this monolithic-type kernels allows an easy restart of the service, if required, without having a huge impact on the other processes of the operative system.
- *Be able to run as a system service*: The server, programmed as a ruby script can easily be set as a system service which can be started or stopped at any time.
- *TCP/IP communication oriented*: The message passing between server/client was programmed over an internationally known standard protocol supported by libraries on many programming languages.
- *Easy modification and maintainability*: Being the code written on a simple language as ruby, the level of abstraction required to modify it is relatively simpler than the one required if the Server was written on other languages where complex data structures and castings are required.
- *Network file transfer optimised*: The third party package *rsync* was chosen due to its stability and optimality for file transfer and keeping up to date copies of a source folder over network systems.

4.2 Development requirements for the client

The following initial list of requirements for the Client summarise the perception and ideas that will be required for this development. In front of each requirement, the solution proposed to suffice this need is presented.

- *Stability*: Programmed over C++ and running on an extensively tested third party package called Paraview [15], the client can guarantee if properly developed the required stability levels for this kind of applications.
- *Have a powerful graphical engine for different purposes*: ParaView, an open source third party package was cho-

sen as the graphic engine for this client. Of relative easy modification once its structure is understood, the plugin programming option became a powerful tool for developing the client.
- *TCP/IP communication oriented*: The message passing between server/client was programmed over an internationally known standard protocol supported by libraries on many programming languages.
- *Easy modification and maintainability*: Paraview code is supported by a large community which can provide insight into adding further functionality and supporting the code.
- *Network file transfer optimised*: The third party package *rsync* was chosen due to its stability and optimality for file transfer and keeping up to date copies of a source folder over network systems.
- *OS portable*: The client was designed as a set of plugging running under Paraview. Paraview is developed over multiplatform packages. Running the client on Windows, Mac or Linux based systems implies that the plugging have to be compiled for each platform supported by Paraview.
- *Compatible with different I/O Devices*: The plugin architecture supports the usability of different I/O devices such as wii-motes, gamepads, and others, through the implementation of different control structures using the VRPN library embedded on a Paraview plugin compatible with the client.

The client application should be able to perform the following operations for the user to interact with the server:

- **Connect/Disconnect** By means of this functionality,the user can start/stop the communication with the VWT server located at the a given machine through an established port. The user should always identify himself using an *Username*.
- **List Simulations** Lists the available simulations the user can start interacting with. The status of each simulation(currently running/stopped) should be available for the user to query.
- **Create Simulation** This functionality allows the user to create a new simulation on the server side.
- **Get Online Users** Lists the users currently logged on the server.
- **Get User Viewpoint** allows to interactively obtain the current view point of any user logged on the VWT application for a given simulation.

4.3 Running the scenario

Once the scenario has been properly established, the user must confirm this action and start the calculations. After the solution processing begins, the Client/Server communication and interaction described on Sect. 3 starts. The user must explicitly confirm the scenario setup, ordering the server to execute the desired command. In case the user desires to perform any change on the scenario during the solution, he is totally free to modify the scenario to his/her will and to refresh the scenario inside the solver. To change the case being solved, the user must explicitly once again tell the server that there is a modification on the scenario.

4.4 Using an already existent Scenario

Any already existent simulation or VWT scenario can be loaded by the client. To load an already existent simulation, the user must first select a valid simulation from the *Simulation list*, inform the server that he will be working on it and then start the setup/modification of a new or existent scenario.

After the simulation is modified, the client will send all the modification requirements to the server, and it will start sending all the information required by the user to perform the steering and postprocessing. This operation requires an amount of time proportional to the bandwidth and the amount of data transferred.

4.5 User feedback

Two tests were conducted upon the implemented VWT platform:

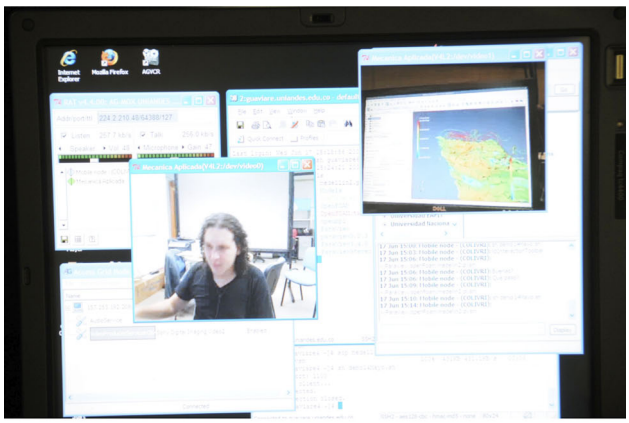
4.5.1 Large model visualisation steering

A large model of the buoyant winds on the Aburra Valley in Colombia (2.6 million cells approx.) was used as dataset for interactive postprocessing between two groups of people on different locations. The first group, located at Eafit University (Medellin, Colombia) used conventional screens and mouse as interaction means, while the second group, located at Los Andes University (Bogota, Colombia) used a large format screen and remote controllers (wiimote type) as interaction means. Both groups were connected via Access Grid [18] conference using audio and live video feeds as intercommunication. Both sites are connected via the RENATA (Colombian academic network, at 30 Mbps at that time). No simulation was run and the platform was only tested as means to set up a remote collaborative environment meant for discussion.

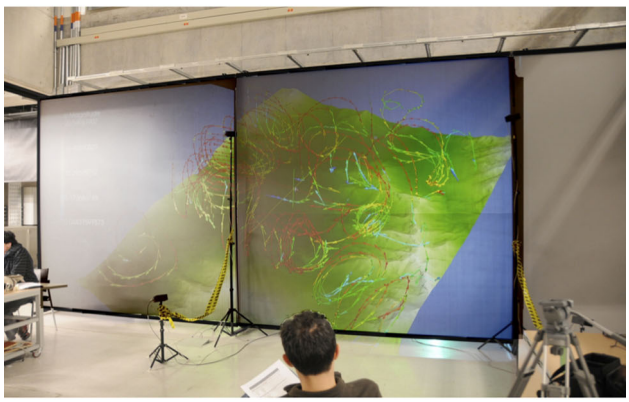
Once the platform was set up on both sites and the dataset was shared, an initial discussion topic was established and for 30 minutes an interactive chat between both parties through the video conference and the same simulation dataset was held. Figure 5 presents the setup of the interaction devices used on both sites.

As a result of the previous test, we can make the following observations:

- Fluent and stable communication between client and servers.
- No locking points were found during the experiments.
- User interaction via remote controls was successful, but still requires more expertise on the user side.



(a) EAFIT University interaction setup



(b) Los Andes University interaction setup

Fig. 5 Interactive post processing test

- Once the dataset is located on both clients, bandwidth is not a big issue.
- Knowledge generation through discussion on different sites is highly enriched by the interactive experience. Communication skills are still key to guarantee a fluid discussion.
- The initial setup of the platform requires specialised personnel, while the interaction and discussion has proven that familiarity with similar tools is only a factor that ease up the initial approach to the tool. Once familiar with the environment, the user is free to explore the environment on any way he desires.

4.5.2 Simulation steering

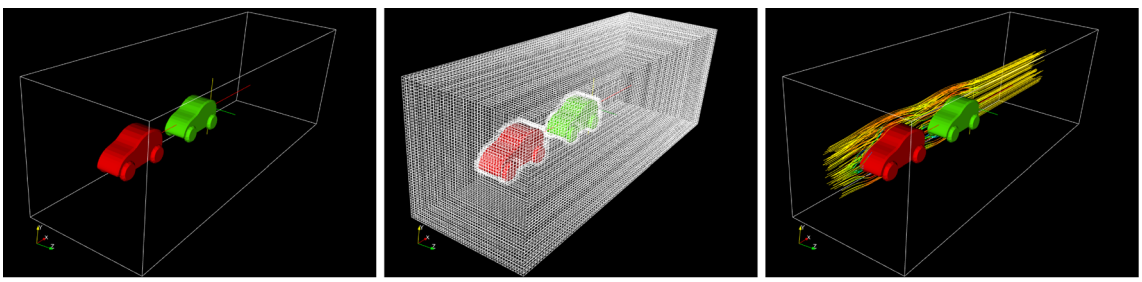
A simplified model of a toy car, formed by 715 facets, was used as a test model for a VWT scenario. The CFD domain around the body was initially subdivided into 150000 hexahedral cells. The simulation was remotely steered by two groups of people at different locations (EAFIT and Los Andes Universities).

Both parties, using Access Grid as video and audio conference system, joined efforts for coordinately steer the simulation. Figure 6 presents the initial state of a simulation that contains two bodies (Red and Green Cars) originally oriented on the same direction. Inside the same figure, the initial finite volume mesh used to solve the case can be seen and to its right, the streamlines that represent the solution can be seen as well.

After a brief discussion, it was required to change the orientation of the Green Car (45 degrees rotation respect to one of its principal axes). The operation was handled by the team at EAFIT and the instructions sent to the server. The simulation was steered and successfully visualised by both clients. Figure 7 shows the result of the steering operation.

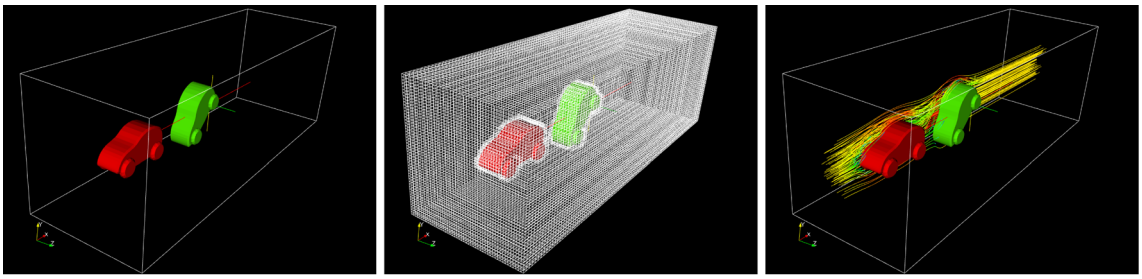
Further testing was performed on the same example using the EAFIT network with 2, 3 and 4 clients, displaying an stable and functional behaviour on all tests.

As a result of the previous test, we can make the following observations:



(a) Initial Scenario **(b)** Initial Scenario Mesh **(c)** Initial Scenario Solution

Fig. 6 Original scenario example



(a) Modified Scenario **(b)** Modified Scenario Mesh **(c)** Modified Scenario Solution

Fig. 7 Modified scenario solution example

- No locking points were identified during the steering. The data exchange worked as expected between client and server and between clients.
- The preprocessing times were fit enough to guarantee a relatively fast simulation setup. The initial required parameters were intuitive enough for the group of users to setup the scenario and recreate the phenomenon.
- Data exchange becomes the bottleneck of this schema due to the large datasets being transferred to the clients. Different schemes than TCP should be tested in order to optimise dataflow.
- Solution time can be an issue, specially when large resolution meshes were used to feed the Finite Volume Solver used in the experiment.
- The users had to be properly trained in CFD practices for properly steering the simulations. The system is aimed at designers, engineers and scientist. Non trained users found difficult to understand the parameters and their physical meaning.

5 Conclusions

A collaborative workspace for CFD simulations/training represent significant opportunities to international (and maybe remote) collaboration specially given the relatively low resources that have to be spent to setup a system. The present implementation uses grid infrastructure for the solver and clients distributed over the network. This architecture has provided useful information about the way users interact with simulation. Message passing time is key to guarantee stability and the depth of the interactive experience. The Fixed-Grid preprocessor is a fast and reliable discretization method for 3D domains where the detail of the features of the object is not the main goal. The discretization is automatically calculated without requiring any user-based expertise or work. Domain decomposition allows this algorithm to be parallelizable and the computational load to be balanced between nodes. Defeaturing may present an advantage for both speed and for providing hints about what the shape might become during the CAE analysis. The solver developed uses the Finite Volume Method. Although it is stable and accurate for solving CFD phenomena, it has show not to be the best choice in terms of minimising solution time. The same basic architecture can be extended to simulate different phenomenon apart from fluid mechanics. Finally, streaming video rendered on the Server machine seems to be a suitable way to avoid sending large data blocks over the network on clients which posses low computational resources and low bandwidth.

Acknowledgments This research was supported by a RENATA reseaarch Grant 2006, from the Ministry of Education and COLCIENCIAS, Colombia. Also supported by EAFIT University, Los Andes University and the University of Alberta.

References

1. Bazilevs, Y., Hsu, M.C., Bement, M.: Adjoint-based control of fluid-structure interaction for computational steering applications. International Conference on Computational Science. Proc. Comput. Sci. **18**, 1989–1998 (2013). doi:10.1016/j.procs.2013.05.368. http://www.sciencedirect.com/science/article/pii/S1877050913005115
2. Billinghurst, M., Kato, H.: Mixed reality—merging real and virtual worlds. In: Preceedings of the First International Symposium on Mixed Reality. Springer, Berlin (1999)
3. Bordegoni, M., Ferrise, F., Ambrogio, M., Caruso, F., Bruno, F.: Data exchange and multi-layered architecture for a collaborative design process in virtual environments. Int. J. Interact. Design Manuf. **4**(2), 137–148 (2010)
4. Borrmann, A., Wenisch, P., van Treeck, C., Rank, E.: Collaborative computational steering: principles and application in hvac layout. Integr. Comput. Aided Eng. **13**(4), 361–376 (2006)
5. Brodlie, K., Wood, J., Duce, D., Sagar, M.: gviz: Visualization and computational steering on the grid. In: Cox, S.J. (ed.) Proceedings of the UK e-Science All Hands Meeting 2004, pp. 54–60 (2004)
6. Bryson, S., Levit, C.: The virtual wind tunnel. IEEE Comput. Graph. Appl. **12**(4), 25–34 (1992)
7. Ferziger, J.H., Perić, M.: Computational Methods for Fluid Dynamics, 3rd edn. Springer, Berlin (2002)
8. Flanagan, D., Matsumoto, Y.: The ruby programming language. 'O'Reilly Media, Inc (2008)
9. Foulser, D.: Iris explorer: a framework for investigation. Comput. Graph. **29**(2), 13–16 (1995)

10. García, M.J.: Fixed Grid Finite Element Analysis in Structural Design and Optimization. PhD. Thesis The University of Sydney, Sydney, Australia (1999)
11. Garcia, M.J., Duque, J., Henao, M., Boulanger, P.: Paravoxel: a domain decomposition based fixed grid preprocessor. Int. J. Comput. Methods (IJCM) (2014, submited)
12. Garcia, M.J., Steven, G.P.: Fixed grid finite elements in elasticity problems. Eng. Comput. **16**(2), 145–164 (1999)
13. Grudin, J.: Cscw: History and focus. Computer-Los Alamitos **27**, 19–19 (1994)
14. Haber, R.B., McNabb, D.A.: Visualization idioms: a conceptual model for scientific visualization systems. Visual. Sci. Comput. **74**, 93 (1990)
15. Henderson, A.: Paraview guide, a parallel visualization application. http://www.kitware.com/products/paraview.html (2007)
16. Jasak, H., Jemcov, A., Tukovic, Z.: Openfoam: A c++ library for complex physics simulations. In: International workshop on coupled methods in numerical dynamics, pp. 1–20 (2007). URLhttp://powerlab.fsb.hr/ped/kturbo/openfoam/papers/CMND2007.pdf
17. Kreylos, O., Tesdall, A., Hamann, B., Hunter, J., Joy, K.: Interactive visualization and steering of cfd simulations. In: Proceedings of the symposium on Data Visualisation 2002, pp. 25–34. Eurographics Association (2002)
18. Laboratory, A.N.: The access grid. Web (2012). http://www.accessgrid.org/
19. van Liere, R., Mulder, J.D., van Wijk, J.J.: Computational steering. Future Generation Computer Systems **12**(5), 441–450 (1997). doi:10.1016/S0167-739X(96)00029-5. http://www.sciencedirect.com/science/article/pii/S0167739X96000295
20. Mulder, J., van Wijk, J.: 3d computational steering with parametrized geometric objects. In: Nielson, G.M., Silver, D. (eds.) Visualization '95 (Proceedings of the 1995 Visualization Conference) pp. 304–311 (1995)
21. Oh, D.: The java virtual wind tunnel—a two dimensional computational fluid dynamics simulation. http://raphael.mit.edu/Java/ (2001)
22. Rank, E., Borrmann, A., Düster, E., Treeck, C.V., Wenisch, P.: Computational steering: towards advanced interactive high performance computing in engineering sciences. In: 8th. World Congress on Computational Mechanics (WCCM8). International Association for Computational Mechanics (2008)
23. Schroeder, W.J.: The visualization toolkit user's guide: updated for version 4.0. Kitware (1998)
24. Shannon, C.: A mathematical theory of communication. Bell Syst. Tech. J. **27**, 379–423 (1948)
25. Shannon, C.: Communication in the presence of noise. Proc. IRE 37 (1949)
26. Taylor II, R.M., Hudson, T.C., Seeger, A., Weber, H., Juliano, J., Helser, A.T.: Vrpn: a device-independent, network-transparent vr peripheral system. In: Proceedings of the ACM symposium on Virtual reality software and technology, pp. 55–61. ACM, New York (2001)
27. Tridgell, A., Mackerras, P., et al.: The rsync algorithm. Tech. Rep. TR-CS-96-05, The Australian National University (1996)
28. Weller, H., Tabor, G., Jasak, H., Fureby, C.: A tensorial approach to computational continuum mechanics using object orientated techniques. Comput. Phys. **12**(6), 620–631 (1998)
29. Wenisch, P., Treeck, Cv, Borrmann, A., Rank, E., Wenisch, O.: Computational steering on distributed systems: indoor comfort simulations as a case study of interactive cfd on supercomputers. Int. J. Parallel Emerge. Distrib. Syst. **22**(4), 275–291 (2007)