European Conference on Computational Fluid Dynamics
ECCOMAS CFD 2006
P. Wesseling, E. Oñate and J. Périaux (Eds)

AN ADVANCED COLLABORATIVE INFRASTRUCTURE FOR THE REAL-TIME COMPUTATIONAL STEERING OF LARGE CFD SIMULATIONS

Pierre Boulanger*, Manuel Julio Garcia†, Curtis Badke*, and Jeff Ryan*

*University of Alberta
, Department of Computing Science
, Edmonton Alberta, Canada
e-mail: pierreb@cs.ualberta.ca
web page: http://www.cs.ualberta.ca/ammi/
†EAFIT University,
Department of Mechanical Engineering
Medellin, Colombia
e-mail: mgarcia@eafit.edu.co

Key words: Fluid Dynamics, Grid Computing, HPC, Computational Steering

Abstract. *Advances in computer processing power and networking over the past few years have brought a significant change to the modelling and simulation of complex phenomena. Problems that formerly could only be tackled in batch mode, with their results visualized afterwards, can now be monitored whilst in progress using graphical means, in certain cases it is even possible to alter parameters of the computation whilst it is running, depending on what the scientist sees in the current visual output. This ability to monitor and change parameters of the computational process at any time and from anywhere is called computational steering. By combining this capability with advanced communications tools, like the Access Grid, over high-speed network it is now possible for a group of scientists located across various continents to work collaboratively on simulations allowing them to compare ideas and to share their experience. In this paper, we present such an advanced collaborative computational steering environment specialized to solve CFD problems.*

1 INTRODUCTION

Advances in computer processing power and networking over the past few years have brought a significant change to the modelling and simulation of complex phenomena. Problems that formerly could only be tackled in batch mode, with their results visualized afterwards, can now be monitored whilst in progress using graphical means. In certain cases it is even possible to alter parameters of the computation whilst it is running, depending on what the scientist sees in the current visual output. This ability to monitor and change parameters of the computational process at any time and from anywhere is called computational steering. By combining this

capability with advanced communications tools, like next generation video conferencing systems, it is now possible for a group of scientists and engineers located across various continents to work collaboratively on simulations, allowing them to compare ideas and to share their experience. This is a key advance as the notion of a scientist and an engineer working alone in his laboratory is disappearing, as problems get larger and more complex. As demonstrated in many instances in the past, new multi-disciplinary teams of experts working collaboratively is the only way to solve today's complex science/engineering problems. In this paper, we present such a system in the context of CFD simulation and illustrate some results obtained so far.

2 LITERATURE REVIEW

The advent of the Grid Computing paradigm can now provide distributed visualization and the close coupling of simulations and visualizations in a steering environment. One can find an excellent review of various systems in Brodlie et al [21] [22].

Distributed collaborative visualization at its most basic level is composed of a pre-generated visualization where users can share the control of the visualization method and parameters one at a time. In a true collaborative environment users are also linked by video-conferencing. For example, by adding to the Access Grid a SGI Vizserver session, WestGrid researchers were able to create a very effective collaborative environment [1].

As described previously computational steering requires that an operator must be able to manipulate simulation parameters by changing them directly in the rendering screen. There are major obstacles to achieve this capability, as one need to be able to reverse the transformation of the data as it passes from simulation to the image. To overcome this difficulty a variety of approaches are proposed in the literature. Software such as AVS Express [2] and IRIS Explorer [15] both use an augmented dataflow model capable of supporting image probing based on feedback loops where the geometrical position of a virtual probe queries the input data and returns an interpolated value at the required point. In other systems such as ParaView [8] the display of streamlines and particle trajectories require seed points, which are readily generated by placing a rake or bar in the image. In many of these systems, iso-surfaces and contour lines can be displayed by placing a data probe in the image from which the software generates the surface or the lines joining all points of iso-value.

Another example of this kind of functionality can also be found in [17] and in Amira [3], one of the most recent additions to the commercial visualization software toolkits. Amira supports a set of visualization components such as cutting planes; streamline seed point initialization and volume segmentation, all accessible directly via the image. Felger and Schröder [31] considered the problem of image interaction in dataflow systems and suggested the separation of the visualization pipeline into two components: the visualization input pipeline (VIP) and the visualization output pipeline (VOP). The goal of the VIP was to reverse the effect of the VOP, the VOP being equivalent to the original, output-oriented visualization pipeline of Haber and McNabb [27]. This principle of a reversible pipeline was based on three techniques: the inverse function method, the look-up method, or the listening method. The first method provides a unique one-to-one mapping between an element of the input set and an element of the output

set, an inverse module can be created which recreates the input from the output. In the second method the VIP module 'remembers' all the incoming and outgoing data of the VOP module and performs a look-up operation to find the required data. In the last method the VIP module forces the corresponding VOP module to re-execute, picking up the input data when it detects output data matching that needed.

Mulder and van Wijk [24] took a different route, adopting a modular approach where, as opposed to a dataflow architecture, it was targeted specifically at computational steering. They used 3D objects that were parameterized by data variables associated with the simulation. In their design, the PGOs (Parameterized Graphical Objects) are used both to visualize the output of the simulation and to steer the parameters by manipulating the objects' characteristics.

The closest system to the system proposed in this paper is the RealityGrid [9] project aiming at studying how scientists in the field of condensed matter, material sciences and biological sciences can effectively use distributed computing and visualization resources using the ICENI framework. The ICENI [6] framework is capable of creating high-level abstractions for scientific computing which will allow users to construct and define their own applications through a graphical composition tool that is integrated with distributed component repositories. The project aims to deliver this environment across a range of platforms and devices on the grid using a scheduling service. The ICENI framework is being implemented in Java and JINI, but is able to inter-operate with the Open Grid Services Architecture (OGSA). Similar to our simulation server, the ICENI framework is used to link a visualization client and server, to pass data to a server running a simulation. The visualization server sends the data off to a renderer (currently based on VTK), which can then send the graphical output back to the visualization client based on OpenGL remote rendering, or on Chromium.

Another computational steering project similar to ours is called gViz [20]. This project is funded by the UK e-Science Core Program, and has two main targets: first, to grid-enable two existing visualization software packages, IRIS Explorer and pV3, in order to bring visualization as early as possible to users of computational grids; second, to develop some longer term thinking on distributed and collaborative visualization, where XML languages are used to describe visualization data, and the visualization programs themselves.

3 COLLABORATIVE STEERING ENVIRONMENT

One of the goals of this project is to explore how advanced collaboration technologies will allow scientists and engineers to collaborate with each other easily, efficiently, and effectively in the context of CFD simulation. More specifically, we are exploring how to maximize the Quality of Experience (QoE) of the end user over a wide range of technology platforms. Critical to this work is the fact that QoE is a human-centred measure not a technology-centred measure. Thus, we want to maximize a human perception based on a rich and diverse set of factors. These factors include, but are not limited to: networking (broadband, wired, and wireless), interaction (3D tracking, touch sensitive devices, voice recognition, mouse, keyboard), and display (immersive displays, wall, table top, desktop, laptop, tablet, PDA). Fundamental to this project is the creation of an architecture to support advanced collaborative environments in the

context of collaborative visualization and computational steering for large CFD problems. This new software architecture is an open one, allowing for the extensible creation of collaboration services and the ability to deploy those services on a wide range of heterogeneous technology platforms as illustrated in Figure 1 and Figure 2. In those systems, the two visualization and steering environments are located at Simon Fraser University (SFU) in Vancouver and at the University of Alberta in Edmonton. The approximate distance between the two cities is 2000 Km. The sites are linked by a 1Gb/s broadband optical network that is part of the WestGrid infrastructure as illustrated in Figure 3. WestGrid is a western Canada consortium on HPC. One can find more information about WestGrid at http://www.westgrid.ca.

In this particular configuration the super-computer, a 256 CPUs SGI Origin 3900, located in Edmonton is used to compute the CFD solutions using an open source software called OpenFOAM which was optimized for an SGI environment. The visualization computer, a 10 graphic pipes SGI computer with 64 Giga-bytes of memory is located at SFU in Vancouver where a visualization program renders each steps of the simulation process. In the following sections, we will describe the various elements of this complex collaborative system.

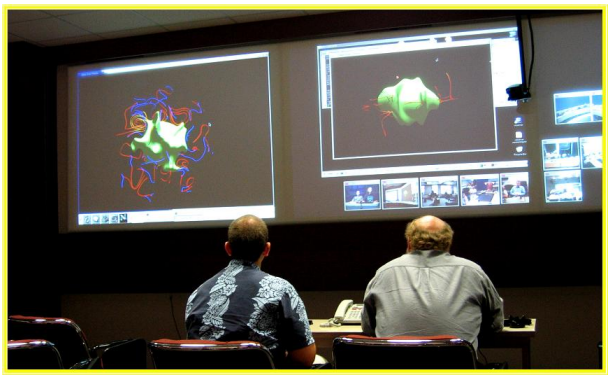


Figure 1: Computational steering environment using AccessGrid at the University of Alberta.

3.1 Simulation Server

One of the critical components for usability of collaborative simulation on a grid is to design a visualization/simulation system in such a way that it can run relatively independent of the original simulation code time scale. In particular, the system architecture should allow three-dimensional (3D) rendering of scalar and vector fields to occur independently of the data production process. This means that data production delays should not cause delays in manipulation of the visual representation, such as object rotation and the exploration of vector fields. In addition to this time constraint, we also need to engineer this system to allow users of a grid infrastructure to get access to advanced visualization without the need for major code modifications.

The main components of the proposed architecture (see Figure 4) are:

Figure 2: Computational steering environment using AccessGrid at SFU.

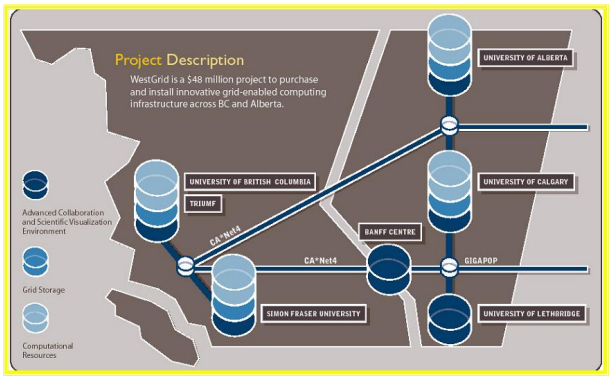


Figure 3: WestGrid high-performance computing facilities.

- A simulation program: the code that need to be computationally steered;
- A simulation controller: responsible for controlling various services requested by the clients and by the simulation process;
- A client manager: who is responsible of offering to each client: simulation, filtering, and graphical services;
- A client display program: who is responsible of connecting to the server and to display to the user the most up-to-date simulation results.

Here the simulation controller directs which data coming from the simulation will be processed. The simulation services converts simulation output for further processing by the visualization program located at each client site. The subsystems 1), 2), and 3) run on a large

multiprocessors shared memory machine or on clusters distributed along the grid whereas subsystem 4) may be run on a remote PC or high end workstation. The data transfer from the large shared memory machine where the simulation server reside and the clients graphic computers is facilitated by data compression, which is performed by an extra service offered by the system.

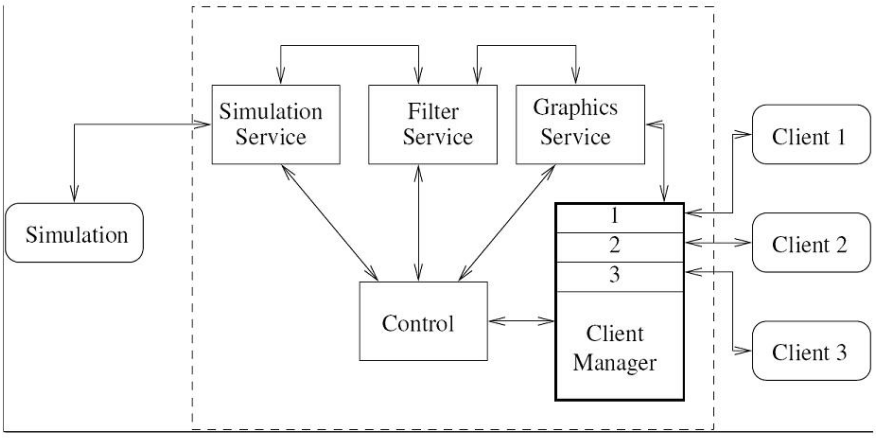


Figure 4: Proposed simulation client/server architecture.

A proof of concept of this architecture was implemented last summer for an earth dynamo simulation code and was demonstrated live at iGrid2005 conference in San Diego California [4]. This earth dynamo simulation code is a modified version of Magic2 [32]] used by Dr. M. Heimpel at the University of Alberta's Department of Physics solving magneto-hydrodynamics equations to simulate earth magnetism. As part of this proof-of-concept, a first version of the solution server running a Magic 2 simulation was running on a 256 CPUs SGI machine located at the UofA. A display client capable of running on the VizRoom three-pipes SGI computer and on a local PC was also developed. For the purpose of the experiment some of these clients were located in San Diego connected on the TeraGrid network using a User Controlled Light Path (UCLP). This demonstration was part of the Global Lambda Visualization Facility at iGRID2005 [12]. Even though this first version was minimal in functionality, it did demonstrate conclusively that such a server is one of the building blocks of an advanced collaborative visualization environment for simulation. The system has the following advantages:

- It is truly a scalable system: the server can run on PC clusters as well as on large shared memory super-computers;
- The client can run on high-end SGI as well as on low-end PC graphic machines;
- It is easy to integrate client to high-end graphics library such as SGI's Performer as well as to OpenGL code;

- Allow independent or synchronized visualization by various clients from various viewpoints;
- The system requires minimal modifications to the original simulation code to be integrated in this framework.

As an addition to this system, we are now adapting the same code to a general CFD solver called OpenFOAM [7]. The OpenFOAM (Field Operation and Manipulation) software package can simulate in parallel anything from complex fluid flows involving chemical reactions, turbulence and heat transfer.

The core technology of OpenFOAM is a flexible set of efficient C++ modules. These are used to build an archive of solvers, to simulate specific problems in engineering.

The new architecture for this so called Virtual Wind Tunnel is illustrated in Figure 5. In this system a CAD model represented as a triangulated mesh is provided to the system and transformed into a volumetric mesh by a parallel volumetric mesh generator capable of producing 2 million volumetric elements per second on a 8 CPUs computer. Following the production of the volumetric mesh, the system add boundary conditions to the mesh and then saved the attributed model in OpenFOAM format on a share disk with the solution server computer using CXFS file management system. OpenFOAM running on a large HPC machine (256 CPUs shared memory system) located to a remote location then read this file and solve the CFD solution for the various time steps requested by the users. The OpenFOAM simulator is controlled by the simulation server allowing various clients to connect to the simulation and to share their results.

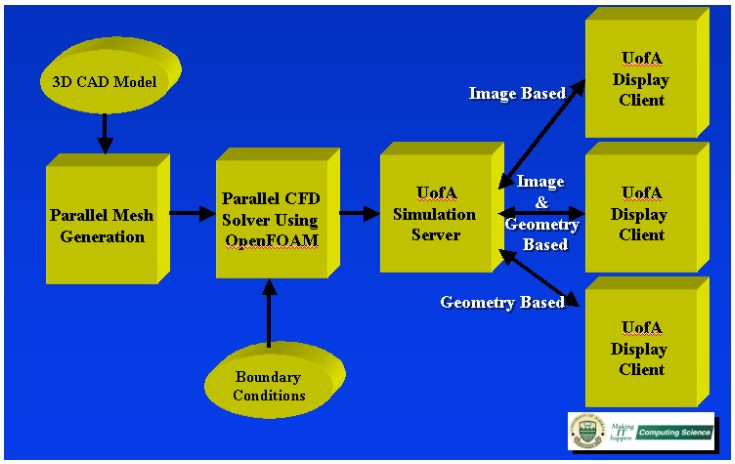


Figure 5: Real-time CFD solver using UofA Simulation Server and OpenFOAM.

In order to illustrate the concept of computational steering in this new environment, we have tested OpenFOAM on a real-life situation where low altitude winds can be predicted using NASA SRTM Digital Terrain Model and CFD.

4 A TEST CASE: LOW ALTITUDE WIND STUDIES USING NASA SRTM DIGITAL TERRAIN MODEL AND CFD

Standards and codes of practice for low altitude wind simulations typically assume terrain to be of homogeneous roughness. In order to deal with real terrain roughness many authors [11] use explicit corrections for specific topographies such as hills or escarpments. For more complex situations many researchers use boundary layer wind tunnel testing to get wind speed and flow distributions. With the fast evolution of computers and algorithms, as well as the availability of high resolution digital terrain models (DTM) across the world, numerical wind simulation or Computational Fluid Dynamics (CFD) Models has become very attractive as they provide a low cost alternative to evaluate wind effects in low altitudes.

Standard computational modelling of the wind is made through the solution of the Navier-Stokes equation,

$$\frac{\partial \rho \vec{u}}{\partial t} + \nabla \cdot (\rho \vec{u} \vec{u}) - \nabla \cdot \mu \nabla \vec{u} = -\nabla p + \rho \vec{g} \, , \tag{1}$$

where $\vec{u}$ is the velocity vector, ρ the density, μ the viscosity and $\vec{g}$ the gravitational acceleration vector. Solution is typically implemented using the Reynolds Average Navier Stokes (RANS) or in some cases where better accuracy is needed Large Eddy Simulation (LES) is used. Turbulence flow can be solved using the $k - \epsilon$ model which is only suitable for the case of homogeneous isotropic turbulence. Discretization of the continuous equations is made by the Finite Volume Element Method which results in a set of algebraic equations [14].

In the past, the main problem for solving the Navier-Stokes equation for low altitude wind simulation was the availably of good DTM of a regions or even a whole country.

On February 11, 2000, the Shuttle Radar Topography Mission (SRTM) was launched into space as part of one of the payload of the Shuttle Endeavor. Using a new radar sweeping technique most of the Earth's surfaces was digitized in 3D in approximately 10 days. SRTM acquired enough data during its mission to obtain a near-global high-resolution database of the Earth's topography. This revolutionary data set can be used to simulate anywhere around the Earth low altitude winds for various atmospheric conditions.

Wind simulation and prediction near ground and developing of high accuracy maps of this winds to help wind farm developers locate optimum areas of high wind energy potential. Wind farms are the fastest growing energy sector in the world today. Pinard et al [26] uses WEST (Wind Energy Simulation Toolkit) to simulate the wind climate of the southern Yukon Territory's mountainous terrain. WEST uses the mesoscale model MC2 (Mesoscale Compressible Community, [10]) and the linearized microscale model MS-Micro/3R [29] to produce a wind map of a large region. The toolkit has been successfully used to simulate the wind energy potential for several regions across Canada and some other regions. However, Pinard's studies over the Yukon region gives errors of about 40% in wind magnitude and direction. A possible cause of the problem is attributed to the rather greatly modified mesoscale terrain used as surface input to MC2. Finally, a review of wind simulations over complex terrain is found in [11]. It presents a comparative analysis among computational simulations and experiment data showing variations of wind flow over hills, valleys and other terrain configurations.

4.1 Format of SRTM Digital Terrain Models

DTM is a representation of elevation data of the earth surface.They are organized in several files which contains latitude, longitude and elevation coordinates of points over the earth surface. One of the most complete collection of high resolution DTM was acquire by the Shuttle Radar Topography Mission (SRTM)[25, 13]. One can see in Figure 6 an illustration of the radar measuring system and in Figure 7 an illustration of the earth coverage performed during the 10 days. Topographic data was collected for dry lands (excluding water masses) between 56 degree latitude South and 60 degree latitude North, covering about 80% of the total surface.

The gathered data was processed and organized in segments covering one degree latitude by one degree longitude. Every segment was saved in a distinct file. Every file is organized as a grid of n rows and m columns of samples whose distance is either one arc second (1/3600 of degree) for United States (SRMT) and three arc seconds (SRMT3) for the rest of the world. In the case of SRTM3 the number of rows and columns is $n = m = 1201$. The rows on North and South border, as well as columns on East and West border, will be identical (and consequently will overlap) to rows/columns of adjacent file borders (see Figure 8).

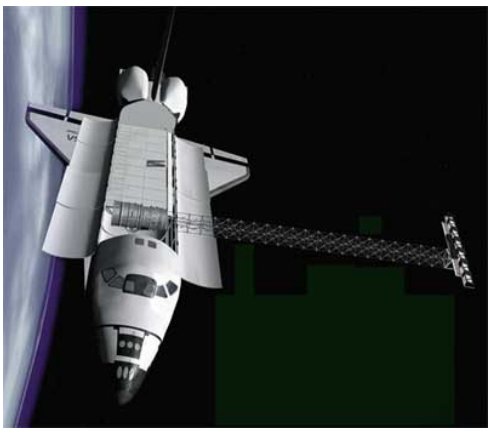


Figure 6: Illustration of the NASA SRTM radar measurement system. (*Image from NASA Website*)

One arc second measured at equator, corresponds approximately to 30 meters, so two sampling points in a STRM3 file are distant roughly 90 meters from each other. This approximation will deteriorate as one move away from the equator in the direction of the poles.

By convention SRTM files names, with .HGT extension, recall the latitude and the longitude of the geometrical sample located in the lower left corner of the segment contained within the file. For example, Figure 8 shows the structure of a file with name N20W100.HGT. It will contain the samples from longitude 20 degree North to 21 degree North and from latitude 100 West to 99 West.

Every sample is represented with a 16 bit signed integer, row ordered. Integer coding is *big endian*, meaning that the first byte read is the most significant and the second is the less

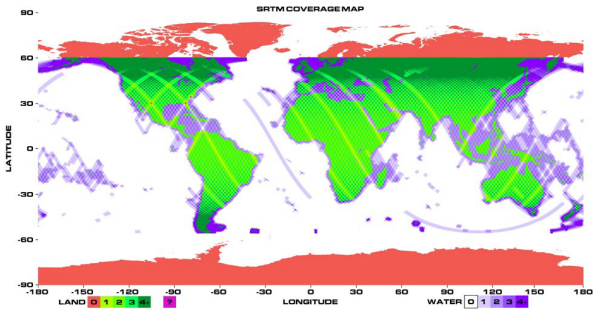


Figure 7: Illustration of the NASA SRTM measured earth coverage. (*Image from NASA Website*)

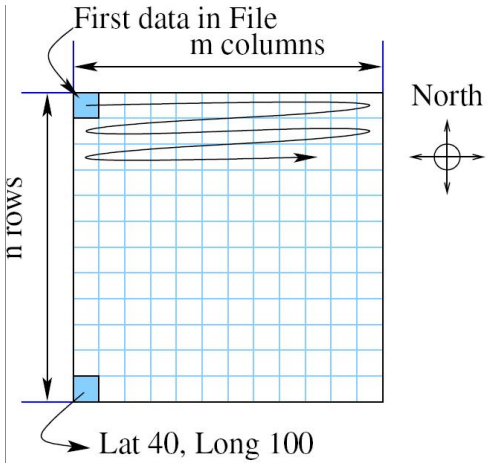


Figure 8: Description of a .HGT file structure. The name file in this case is N20W100.HGT

significant. The number associated with each sample corresponds to its elevation measured in meters with respect to the WGS84 EGM96 geoid (the geoid is a rotational solid used in geodesy, that approximates the shape of the earth better than a sphere or an ellipsoid). The data files are not perfect and some *voids* can be present. They are due to shadows or water reflection creating distortions. Data samples corresponding to these voids are given a conventional code value of -32768.

Processed data can also be retrieved through the Seamless Data Distribution System (SDDS) of the U.S. Geological Survey (USGS) and the EROS Data Center (EDC) [28]. The main feature is that the user can define an area of interest by drawing a box, defining coordinates, or use available templates to cut out the area. They also offer a variety of DTM formats.

4.2 Definition of the CFD Domain for Wind Study

A CFD domain is created from the DTM information and it should covers a large enough portion to guaranty accuracy in the simulation. For low altitude wind simulation a volume of

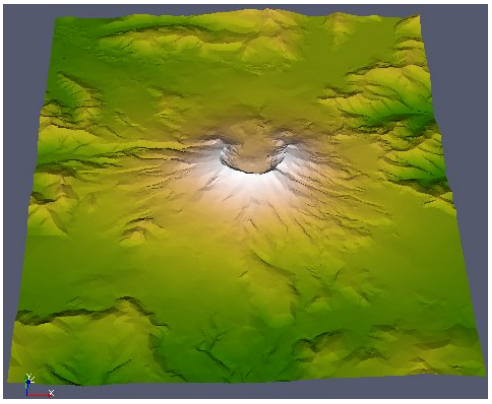


Figure 9: Digital terrain model of mount Saint Helens region produced by SRTM.

air over the surface constitutes the CFD domain.

4.3 Filling Shadow Regions

Data from the SRMT should be completed to generate the domain. Voids can be filled in different ways using several interpolation techniques. Also, information from other sources, like other DTM databases or GPS measurements, can be used to complete the data.

A simple interpolation technique using information from the closest points around a void $\vec{x}$ is used to interpolated the points inside the void. The interpolation function used is defined by:

$$z(\vec{x}) = \frac{1}{n} \sum_{\vec{q}_j \in \Omega_x}^{n} w_j \; z(\vec{q}_j), \quad \Omega_x = \left\{ \vec{q}_j \;\middle|\; ||\vec{x} - \vec{q}_j|| < r \right\}, \tag{2}$$

with w_j a weight factor and $r \in \mathbb{R}$. This weighted average can be further enhance if a least square approximation is taken over a patch $||\vec{x} - \vec{q}|| < r$ around the void [16].

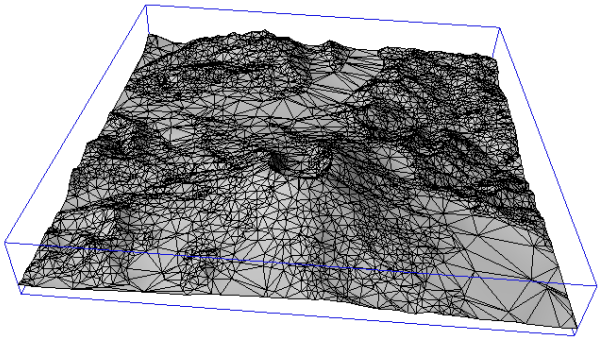
During this project, a new hole filling algorithm was developed using Radial Basis Functions. We are currently testing its capability for DTM applications. One can refer to the paper by Branch et al. [19] for more details.

4.4 From Surface Mesh to Volumetric Mesh for CFD

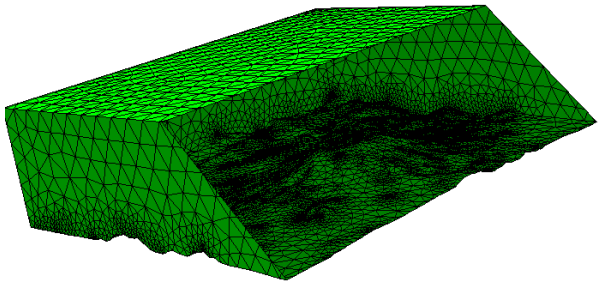
The finished DTM data is transformed into a regular grid of triangular elements, e.g. a 2-manifold with border that represents the terrain surface. The SRMT data files cover an approximate area of 108×108 km. As SRMT3 data is sampled at every 3 arc sec it will contains in total $1'442,401$ equidistant points. This surface can be efficiently decimated preserving the curvature detail by suppressing points at low curvature regions. Depending on the topography of the terrain it can be reduced up to 15 to 10 % of the original data while keeping the approximation error low. In order to do so we used POLYWORKS IMCompress [5] to perform this

decimation. The algorithm eliminates sequentially the triangle that produce the smallest error until an approximation threshold is reached.

CFD domain consist of a volume of air over the terrain surface. It can be generated by projecting vertical walls along the sampled terrain boundary and then intersecting the resulting shell with a horizontal plane at a given height h. The final manifold must be closed and topologically correct (See Figure 10b).



(a) Digital terrain model after decimation using POLYWORKS IMCompress.



(b) The CFD domain created over the decimated digital terrain model.

Figure 10: Decimated mesh and volumetric domain for CFD analysis.

Once a correct superficial mesh of the domain is obtained a volumetric mesh needs to be generated. An ideal mesh for the volume element method consist of hexahedral elements. However, it is very complex to obtain automatically and there are several research groups working in this field. An alternative to structured hexahedral meshes are the so called tetrahedral meshes. They should be used with caution when using volume element method and orthogonal correctors should be used to insure convergence of the CFD solver. In this study, tetrahedral meshes are generated using a parallel mesh generation program based on fixed grid. The fixed grid model is generated by immersing the given structure as a solid model (B-Rep), into a rectangular grid of equal-sized elements. In this way, elements are either inside (I), outside (O), or on the boundary (NIO) of the structure. The material properties for O elements are the ones of a non-interactive medium. That is, its values are significantly less than the ones for the I element

1. SolidModelLoad()
2. FacetsPerRayCompute()
3. NodesPerRayClassify()
4. ElementsClassify()
5. NioGeomCompute()
6. MeshExport()

Figure 11: Fixed Grid Main Meshing Algorithm

material properties. This transforms the problem into a bi-material formulation. NIO elements are constituted by two types of material and therefore their properties are not continuous over the element. Different methods can be used to approximate NIO elements. That includes from treating them as O elements to complex non-continuous domain integration. By computing NIO element properties as averaged, the element-stiffness matrix can be computed as a factor of a standard stiffness matrix which makes the global stiffness matrix assembly a very efficient process. An additional advantage is found when the shape of the structure is changed in response to a previous analysis.

For the case of Fluid Mechanics, the material approximation is not used, but NIO elements are approximated by fitting elements of the same or lower degree than the hexahedra elements. This is, wedges, pyramids and tetrahedral elements are used to represent the subdivision of the NIO elements.

The objective of the preprocessor described in this section, is to compute the mesh resulting after a Fixed Grid is superimposed to a solid, by constructing the elements, determining its type (I, O or NIO) and obtaining the subdivision for the NIO elements depending on the final use of the resulting mesh, which can be for structural or fluid mechanics simulations.

4.4.1 The Fixed Grid Meshing Algorithm

Given that the solid model boundary is represented as a piece-wise linear (PL) surface, represented by two-dimensional simplices, the domain subdivision algorithm is based on ray tracing over the set of simplices. This ray tracing process, as shown in Figure 12 is made by intersecting the rays r_i in $z+$ direction projected from the lowest plane.

Every ray r_i contains a set of intersection points which is used to classify the nodes of the fixed grid as inside, outside or boundary. With the classified nodes, the element classification takes place. The process ends with the NIO elements geometry reconstruction and boundary conditions assignment. At this point the resulting mesh can be exported. Algorithm illustrated at Figure 6 outlines the main steps in the fixed grid meshing process.

With this filter, the number of facets to intersect with every ray is reduced significantly. Figure 12 shows the projection of a given facet over the plane π_0 and the ray that intersects it.

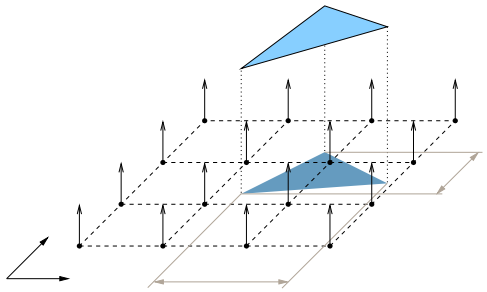


Figure 12: Ray facet intersection testing.

4.4.2 Parallel Implementation

In order to reach real-time mesh computation a parallel scheme was implemented. In the proposed scheme the Server-Client approach was used. In this approach the solid model geometry is loaded in the server and all the clients and every step of the algorithm (see Figure 6) is subdivided over all the CPUs. The information necessary for every step is shares among the processes using a Message Passing Interface.

Message Passing Interface (MPI) is a library specification for message-passing, proposed as a standard by a broadly based committee of vendors, implementors, and users.

The parallel computation bottleneck is in the message passing, this is the reason why the communication requests have to be minimized in a way that every process makes as many operations as possible without sharing information with the others, and in the case of sending information, use a single packet for all the data.

After all the steps in the main algorithm are finished, the information is collected in the server. Because the resulting mesh is structured, only the state of the elements has to be known in order to recover the I elements. For the NIO elements, the information about the intersection points has to be shared among all the processes. For this reason, the less NIO elements in the model, the less data that has to be shared between the server and the clients. A speed of 2 million volumetric elements per second on a 8 CPUs PC cluster was achieved using this parallel processing algorithm.

4.5 Definition of Boundary Conditions

Due to the boxed shape of the domain, boundary surfaces are the terrain surface, the south, north, east and west walls, and the top surface.

Wind velocity at input boundary can be obtained using a simple logarithmic rule that makes use of the surface roughness z_0 for each site, namely

$$u(z_p) = u(z_m)\frac{\ln \frac{z_p}{z_0}}{\ln \frac{z_m}{z_0}} \ , \tag{3}$$

where the wind speed $u(z_m)$ measurement (or modelled) at height z_m is projected vertically to new speed $u(z_p)$ and height z_p. This logarithmic projection is done instead of obtaining the wind speeds directly from the model for the exact heights at which the measurements were made. This approach requires wind measurements to extrapolate the wind profile at the entrance wall.

Fictitious wind scenarios can be created to help understand typical real conditions over a geographic area. In such case several sets of boundary conditions should be analyzed. Wind profiles over input boundaries can be obtained by applying

$$u(z_p) = a\ e^{-\left(\frac{z-c}{b}\right)^2} + m, \tag{4}$$

with a (amplitude), b (wide), c (centre) and m (offset), factors that control the shape of the profile function. Combination of these functions allows to create velocity profiles similar to the one found in nature. Other boundary conditions include:

- Uniform input $\vec{u} = (10, 0, 0)$ m/s;
- Outlet (with fixed pressure $p = 0$ Pa;
- No-slip walls over the terrain;
- Wall functions over the terrain;
- symmetry condition over the top wall $\nabla\vec{u} \cdot \hat{n} = 0$.

It should be noticed that, as well as in a wind tunnel, accurate results are only expected at some distance from the boundary walls.

4.6 Physical Model Used

From all the methods, Direct Numerical Simulation (DNS) produces the most accurate results. It is however, the most expensive of all the solving methods as it maintains small variations of the turbulent flow. Large scale motion is much more energetic than small scale motion. It is responsible for transport of the conservative properties. It has complete information of the flow. Large Eddy Simulation (LES) works with large scale motion while neglecting low scale motion. It is a three dimensional time dependent approach which is computationally expensive but less costly than DNS. As it only model large scale motion it is also less accurate than DNS [23]. The objective of flow simulation is to predict selected properties of the flow at minimum

cost. It is not easy to predict how well each method will work. In many situations RANS methods are successful [11] for modelling of low altitude wind flow over complex terrain.

In this implementation, the low altitude wind over the terrain is simulated using the OpenFoam toolkit. The OpenFOAM distribution contains numerous solvers that includes from basic flows as Laplacian, potential, steady state and transient incompressible flow, non-newtonian flow, compressible flow, transonic, supersonic, multiphase flow, direct numerical simulation (DNS) and large eddy simulation (LES), heat transfer and buoyant flow, among others [30, 18].

For low altitude wind simulation the governing equations are: the mass continuity:

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \vec{u}) = 0. \tag{5}$$

and the momentum conservation equation:

$$\frac{\partial \rho \vec{u}}{\partial t} + \nabla \cdot (\rho \vec{u}\vec{u}) + \nabla \cdot (\mu \nabla \vec{u}) = -\nabla p + \rho \vec{g}. \tag{6}$$

In the $k - \varepsilon$ model two extra equations are needed: one for closure and one for the transport equations for k and ε:

$$\frac{\partial (\rho k)}{\partial t} + \nabla \cdot (\rho \vec{u} k) = \nabla \cdot \left(\left(\mu + \frac{\mu_t}{\sigma_k} \right) \nabla k \right) + P_k - \rho \varepsilon, \tag{7}$$

$$\frac{\partial (\rho \varepsilon)}{\partial t} + \nabla \cdot (\rho \vec{u} \varepsilon) = \nabla \cdot \left(\left(\mu + \frac{\mu_t}{\sigma_\varepsilon} \right) \nabla \varepsilon \right) + \tag{8}$$

$$\frac{\varepsilon}{k} \left(C_{\varepsilon 1} P_k - C_{\varepsilon 2} \rho \varepsilon \right) \tag{9}$$

where μ_t is the coefficient for turbulent viscosity and is linked to the turbulent kinetic energy k and dissipation ε by $\mu_t = \rho C_\mu \frac{k^2}{\varepsilon}$. The various constants of the turbulence model [14] are empirically determined and taken for the $k-\varepsilon$ model as: $C_\mu = 0.09$, $C_1 = 1.44$, $C_2 = 1.92$, $\sigma_k = 1$, and $\sigma_\epsilon = 0.76923$.

To guaranty convergence of the solution an initial field is needed. It can be obtained by solving for potential flow φ over the domain by

$$\nabla^2 \varphi = 0 \tag{10}$$

and solving for $\vec{u}$ from

$$\operatorname{grad} \varphi = \vec{u}. \tag{11}$$

Topology of the terrain is obtained from a DTM as a SRTM or BIL file. The surface is reconstructed and the voids filled producing an optimum 2-manifold with border representing the surface of the terrain. A volume is completed by intersecting a cuboid with this surface so that the result is a box with the lower side replaced by the terrain surface. Boundary conditions are set to the domain and a tetrahedral mesh is generated. This mesh is in turn converted into a OpenFoam format. The partial differential equations are solved and the solution is visualized using the visualization client specialized for CFD applications.

4.7 Low Altitude Wind Flow Over Mount Saint Helens

This example simulates the low altitude wind over Mount Saint Helens in the United States. The coordinates are $46^o11'28''$ N and $122^o11'39''$ W. A view of the terrain after the surface mesh is generate is shown in Figure 9. The CFD domain is generates with a box size of 18838 m in the x (West-East) direction, 17476 m in the y (South-North) direction, and the top face is located at a height of 6000 m. For this section, the minimum height of the terrain is 468 m and the maximum height 2490 m. The problem was solved using a mesh of 262,078 elements. The kinematic viscosity of air was taken as $\nu = \mu/\rho = 18.1 \times 10^{-6}/1.293 = 14.0\ \mu$ m^2/s. The flow was calculated with standard $k\varepsilon$ model as described in the previous section. Figure 13 shows stream lines near the mountain summit.

5 Conclusion

These experiments in collaborative computational steering have shown the a great increase in productivity and insights for fluid mechanics problems can be achieved by those systems. We are not working blind anymore. We can now see on the fly the influence of viscosity, boundary conditions, and many more physical parameters that influence the flow pattern. We can abort solutions when they are not what we want instead of computing for hours to find out at the end that the solution is no good. We can do it in real-time on small problems and step by step when the problem is too big. We have also shown that the decoupling of simulation time with visualization time can give us the illusion of real-time.

One of our ultimate goals is to create a truly interactive virtual wind tunnel system for teaching and design. For large problems the current computer technology is too slow even when we use some of the most powerful super-computer. By code optimization, computer upgrade, and more advanced parallel algorithm we are planning to push this boundary further.

One of the numerous challenges associated to this project is to link collaborative technologies, simulation tool, and design tools into a truly integrated environment. We believe that WestGrid computing and communication infrastructures in addition to new emerging standard for CFD and FEM simulation such as OpenFOAM can make this dream possible.

Acknowledgments

This work was partially funded by Alberta's Informatics Center of Research Excellence (iCORE) as well as by NSERC and ASRA.

REFERENCES

[1] http://agcentral.org/downloads/agvizserver/.

[2] http://help.avs.com/avs5/.

[3] http://www.amiravis.com/.

[4] http://www.igrid2005.org/.

[5] http://www.innovmetric.com/.

[6] http://www.lesc.ic.ac.uk/iceni/index.html.

[7] http://www.openfoam.org/.

[8] http://www.paraview.org.

[9] http://www.realitygrid.org.

[10] R. Benoit, M. Desgagne, P. Pellerin, S. Pellerin, Y. Chartier, and S. Desjardins. The canadian mc2: A semi-lagrangian, semi-implicit wideband atmospheric model suited for finescale process studies and simulation. *Mon. Wea. Rev.*, 125:23822415, 1997.

[11] G. T Bitsuamlak, T. Stathopoulos, F. ASCE, and C B. Numerical evaluation of wind flow over complex terrain: Review. *Journal of Arospace Engineering*, 17(4):135–144, Octubre October 2004.

[12] Jason Leigh et al. The global lambda visualization facility:an international ultra-high-definition wide-area visualization collaboratory. *Future generation computer systems, the international journal of grid computing: theory, methods and applications*, 2005.

[13] T.G. Farr and M. Kobrick. Shuttle radar topography mission produces a wealth of data. *Amer. Geophys. Union Eos*, 81:583–585, 2000.

[14] J. H. Ferziger and M. Perić. *Computational Methods for Fluid Dynamics*. Springer, 3rd edition, 2002.

[15] D. Foulser. Iris explorer: A framework for investigation. *Computer Graphics*, 29.

[16] Manuel J. García. *Fixed Grid Finite Element Analysis in Structural Design and Optimization*. PhD. Thesis The University of Sydney, Sydney, Australia, 1999.

[17] P. Hastreiter and T. Ertl. Fast and interactive 3d-segmentation of medical volume data. *In H. Niemann, H.-P. Seidel, and B. Girod, editors, Image and Multi-dimensional Digital Signal Processing '98*, pages 41–44, 1998.

[18] H. Jasak, H.G. Weller, and N. Nordin. In-cylinder CFD simulation using a C++ object-oriented toolkit. SAE Technical Paper 2004-01-0110, 2004.

[19] F. Prieto J.W. Branch and P. Boulanger. Automatic hole-filling of triangular meshes using local radial basis function. *Proceedings of the IEEE 3DPVT 2006 Conference*, 2006.

[20] D.A. Duce K. Brodlie, J. Wood and M. Sagar. gviz: Visualization and computational steering on the grid. *Proceedings of the UK e-Science All Hands Meeting 2004, editor Simon J. Cox*, pages 54–60, 2004.

[21] J.R. Gallop J.P.R.B. Walton K. Brodlie, D.A. Duce and J.D. Wood. Distributed and collaborative visualization. *COMPUTER GRAPHICS forum*.

[22] J.R. Gallop M. Sagar J.P.R.B. Walton K. Brodlie, D.A. Duce and J.D. Wood. Visualization in grid computing environments. *Proceedings of IEEE Visualization 2004, edited by Holly Rushmeier, Greg Turk and Jarke J. van Wijk*, pages 155–162, 2004.

[23] A. Leonard. Energy cascade in large eddy simmulations of turbulent fluid flow. *Adv. Geophys.*, 18A:237, 1974.

[24] J.D. Mulder and J.J. van Wijk. 3d computational steering with parametrized geometric objects. *In G.M. Nielson and D. Silver, editors, Visualization '95 (Proceedings of the 1995 Visualization Conference)*, pages 304–311, 1995.

[25] National Geospatial-Intelligence Agency (NGA), the National Aeronautics, and Space Administration (NASA). The shuttle radar topography mission (srtm). `http://www2.jpl.nasa.gov/srtm/`, 2000.

[26] Jean-Paul Pinard, Robert Benoit, and Wei Yu. A WEST wind climate simulation of the mountainous yukon. *Atmosphere-ocean*, 43(3):259282, 2005.

[27] R.B.Haber and D.A.McNabb. Visualization idioms: A conceptual model for scientific visualization systems. *In Visualization in Scientific Computing (B.Shriver, G.M.Nielson and L.J.Rosenblum, editors), IEEE Computer Society Press*, pages 74–93, 1990.

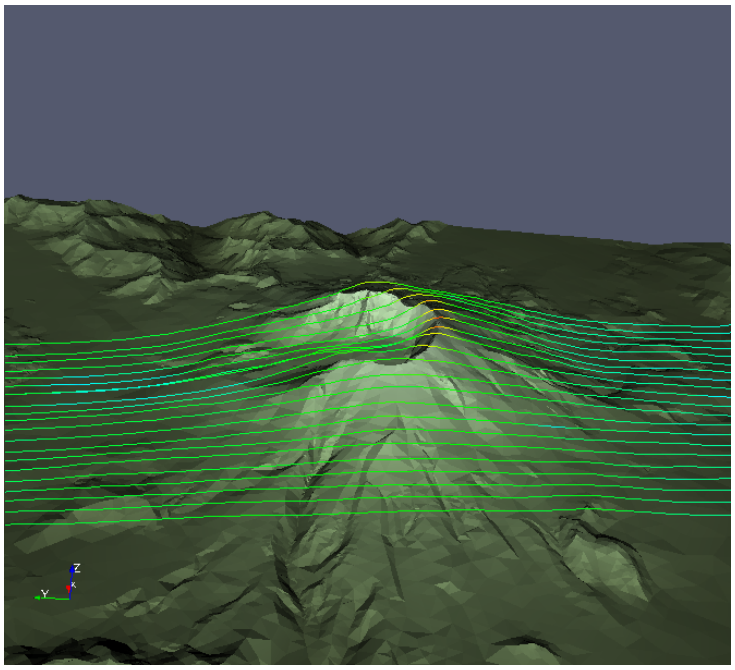
[28] US. Geological service. Seamless data distribution system. `http://seamless.usgs.gov/`, 2003.

[29] J. Walmsley, D. Woolridge, and J. Salmon. Ms-micro/3 user's guide, 1990.

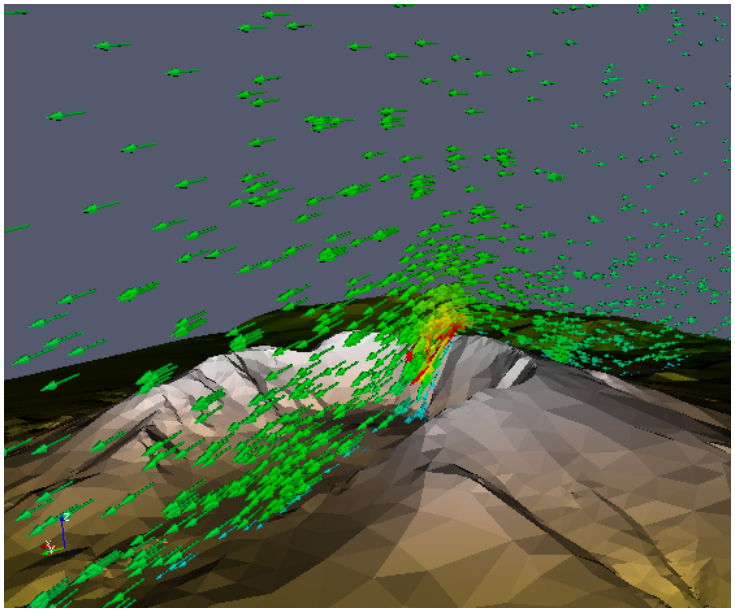
[30] H.G. Weller, G. Tabor, H. Jasak, and C. Fureby. A tensorial approach to computational continuum mechanics using object orientated techniques. *Computers in Physics*, 12(6):620 – 631, 1998.

[31] W.Felger and F.Schroder. The visualization input pipeline - enabling semantic interaction in scientific visualization. *Computer Graphics Forum, Eurographics '92 edition*, 11.

[32] J. Wicht. *Phys. Earth Planet Int.*, 2002.



(a) Flow lines over Mount Saint Helens.



(b) Detail of the flow lines near the summit.

Figure 13: Low altitude velocity wind flow over Mount Saint Helens.