Collaborative and Interactive CFD Simulation using High Performance Computers

Petra Wenisch,
Andre Borrmann, Ernst Rank, Christoph van Treeck
Technische Universität München*
{wenisch, borrmann, rank, treeck}@bv.tum.de

Oliver Wenisch
Leibniz Rechenzentrum München†
wenisch@lrz.de

Abstract

This paper presents a prototype application of a collaborative computational steering system for interactive computational fluid dynamics (CFD) simulations. One or more visualization clients allow the user to interact with the concurrently running CFD simulation. To provide interactivity even for high resolution runs the parallel Lattice Boltzmann-based simulation kernel is performed on a high performance computer (HPC), e.g. the Hitachi SR8000 of the LRZ (Leibniz Computing Center) Munich. This kind of online simulation tool enables engineers to investigate and discuss several case studies interactively to obtain an intuitive understanding of detailed calculations.

1 Motivation

Numerical simulations in the domain of fluid mechanics are nowadays considered to be an important supplement to classical wind tunnel experiments in engineering practice. Typically, these simulations are realized as a batch process consisting of the following interdependent steps: A time-consuming pre-processing step maps CAD data to computational grids and defines boundary conditions, followed by the computation and a post-processing analysis. These steps are usually carried out on different hardware systems and the data is transferred by means of file interfaces.

Unfortunately, the pre-design phase of buildings usually lasts only a short time and later changes to the design incur a dramatic increase in costs. Therefore, an interactive tool is desired to be able to quickly perform several case studies for preliminary investigations, possibly followed by just a few carefully selected simulations with more details. In addition, the building industry is fragmented into numerous involved disciplines working together in a highly cooperative process. The cooperative and interactive CFD application presented here was developed with this kind of procedure in mind.

*Lehrstuhl für Bauinformatik, Arcisstrasse 21, D-80290 München, Germany, http:www.inf.bauwesen.tu-muenchen.de

†Leibniz Rechenzentrum, Barerstrasse 21, D-80333 München, Germany, http:www.lrz.de

2 Computational Steering of CFD Simulations

Computational steering integrates the three steps of pre-processing, computation and post-processing into a single environment allowing for interactive control and modification of the computational process during execution ([1]).
To achieve interactivity, a computational steering application requires the immediate response of a simulation process to user interaction (e.g. adding, removing or replacing objects in a simulated scene). The main requirements for this purpose are fast computation, short-latency communication between user and simulation as well as the rapid processing of modifications due to user interaction.

2.1 Lattice Boltzmann Method and Grid Generation

The Lattice-Boltzmann method (LBM) has emerged as a complementary technique for the computation of fluid flow phenomena. It is an explicitly designed method for solving fluid dynamics problems ([2]). By computing the dynamics of particle densities for a discrete number of velocities and directions at each grid point of spatially discretized scene appropriately, quantities such as mass and momentum are conserved to fulfill the hydrodynamic laws. The Lattice-Boltzmann algorithm computes the *collision* of microscopic, 'virtual' particles and updates the velocity distribution functions within each simulation time-step. Because of the nature of interparticle collisions, this computation can be done for each independent grid point. This leads to the massive parallelization capability of the LBM. Following the collision, the new distribution functions are migrated to their neighboring cells, which is referred to as *propagation*.
Typically, the LBM is implemented on uniform Cartesian grids, thus permitting fast, automatic grid generation. This aspect is an essential requirement of an interactively steerable CFD application allowing modifications of the geometry, e.g., by inserting or deleting fluid obstacles. In our application, the user can load arbitrary triangular, CAD-generated geometries which the grid generator (see [3]) transforms (and transfers) onto a uniform Cartesian grid representation (Fig. 1). The corresponding voxelization algorithm for an optimized grid generation is based on the hierarchical space-partitioning concept of octrees and is parallelized very efficently using OpenMP.

2.2 Parallelization and Optimizing of the Computation Kernel

Special care has been taken to take full advantage of the vectorization and parallelization capabilities of the specialized Hitachi SR8000 hardware ([4]). This system is a pseudo-vector machine with 168 SMP nodes, each consisting of 8 computation CPUs. Eight nodes are available for interactive use.
To parallelize the CFD computation, the fluid domain is divided up into slices along the x-axis and each slice is assigned to one SMP node. Vendor-optimized MPI libraries are used for communication between these SMP nodes. Within one SMP node the main computation loops have been parallelized using compiler directives in the so-called COMPAS[1]

[1] COMPAS: Co-operative Micro Processors in Single Address Space

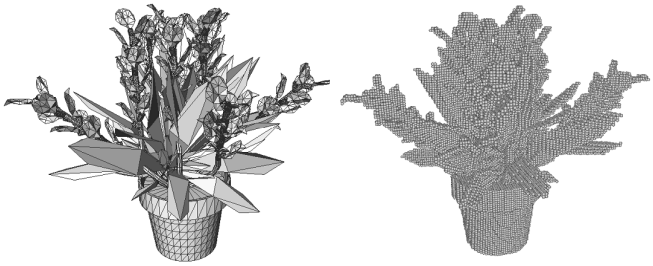

Figure 1: Discretization of a CAD-generated geometry: On the left a plant object's approximation using 30014 triangles is shown facing its discrete representation (30074 voxel, grid resolution 100x100x100) on the right. The voxelization takes only 0.19 seconds on an AMD Dual Opteron with 2.4 GHz.

mode. The powerful vectorization and software pipelining abilities of the SR8000 significantly speed up the computation of these loops . The code has been especially rewritten in order to exploit these capabilities to the full. The data layout has been optimized for the collision step by storing all the velocity distributions on each node contiguously in the memory and integrating the propagation into the collision loop. To enable software pipelining conditional statements have been replaced by equivalent floating point operations, i.e., boolean expressions are mapped onto real-valued coefficient arrays for multiplication. The resulting simplified loop structure enabled the compiler to analyze and optimize the code more effectively. Fig. 2 points out the dramatic increase in performance.

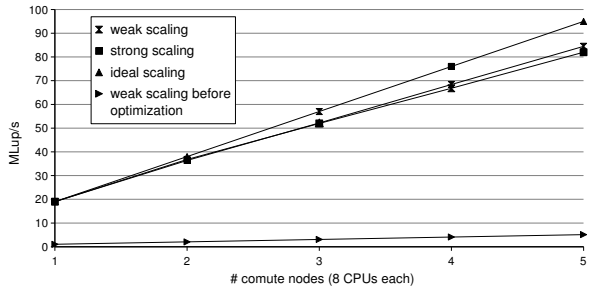

Figure 2: Offline simulations without steering and visualization: The application shows good performance measured in MLup/s (million lattice site updates per second) and high efficiency for strong and weak scaling with 86 and 89 %, respectively. Additionally, the plot visualizes the striking improvement in performance as compared to the non-optimized version of the code.

2.3 Communication

Since computation and visualization are each performed on specialized hardware, a heterogenous hardware set-up is usually required. Therefore, the resulting data have to be sent to the visualization client while data describing the current simulations and fluid parameters have to be forwarded simultaneously to the supercomputer, as well. Since user interactions are spontaneous occurrences, regular communication intervals cannot be applied without loss of performance. Because of these interactions, the result data are only sent at regular intervals as long as no modifications have taken place. The data flow and the main components of the interactive application can be seen in Fig 3. A communicator node (SIM-M) has been introduced on the supercomputer to coordinate communications without interrupting computation. The communication node collects the result data from all computation slaves (SIM-S) and sends it to the visualization workstation as a single message to avoid additional latencies. Incoming data due to user interaction is pre-processed on the communicator node and then distributed to the slaves according to their fluid domain. Introducing this additional node on the supercomputer considerably increases the overall performance of the application, because of the reduction in communication latencies and the overlap of computation and communication. Details of the communication concept and the performance measurements can be read in [5].

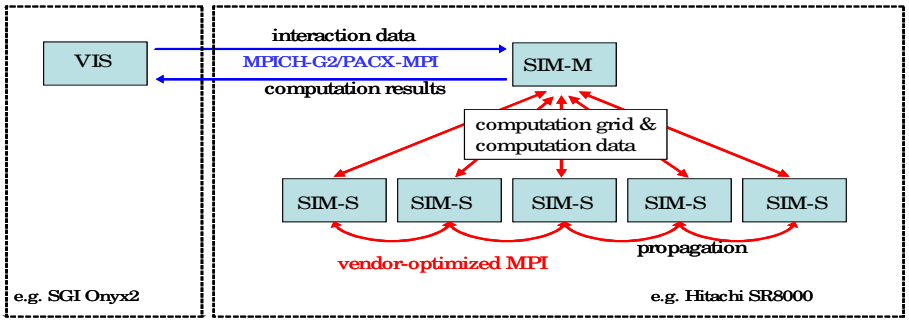

Figure 3: Application scheme showing the visualization (VIS) and the simulation consisting of a master node (SIM-M) and computation slaves (SIM-S). At the LRZ, the VIS process is usually run on a separate graphics workstation, whereas the simulation uses several nodes on the Hitachi SR8000. Within the SR8000, all processes communicate via a vendor-optimized version of MPI. Inter-machine communication between VIS and SIM-M is implemented using either Globus MPICH-G2 ([6]) or PACX MPI ([7]). The simulation side transfers its most recent pressure and velocity fields to the visualization at short intervals. Throughout the ongoing computation, the user is able to analyze these results and even change the scene geometry or simulation parameters, as required.

3 Collaborative Engineering

In order to support the synchronous collaboration between engineers at different locations during the planning of HVAC (Heating Ventilation Air-Conditioning) systems, the interactive application described in 2 has been extended to a distributed collaboration platform which enables the participating engineers to work with the interactive CFD simulation simultaneously. It serves as a distributed multi-user application, i.e., everyone participating in the collaborative session can work interactively with this application by means of an individually configurable human-machine interface.
The basic architecture of the conceived platform consists of a central collaboration server, an arbitrary number of clients and possibly an arbitrary number of simulation servers. Fig. 4 shows these components and the communication paths between them. Each of the components can be run on a different machine, different clients can receive data from different simulation servers and not all of the clients necessarily have to receive simulation data.

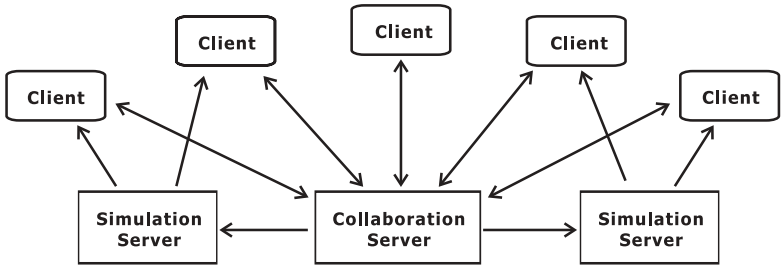

Figure 4: Overview of the multidisciplinary collaboration platform. Clients may contact different servers, or no simulation server at all; however, they need to have access to a collaboration server to report changes or be notified about changes to the application set-up which may have been initiated by other users. The collaboration server, of course, has to forward incoming modification signals to the simulation.

The collaboration server has to perform several tasks such as the managing of registered users, their roles and rights, managing of the common model, and controlling concurrency. It also has to manage the simulation servers, their locations, their start-up and steering parameters. Each of these tasks corresponds to a dedicated module in the collaboration server (see Fig. 5). The core of the collaboration server is the model management module. The common model is basically a geometric model to which additional non-geometric information can be attached. In order to support the evolution of the capabilities of available simulators, the structure of the semantic part is not hardwired, but can be extended by means of a meta-model ([8]).
In the case of our CFD application, the common model represents the obstacles in the fluid domain and the fluid domain hull. The boundary conditions (e.g. pressure, velocity or temperature conditions), which are managed as semantic data attached to the geometric objects, are of special importance.

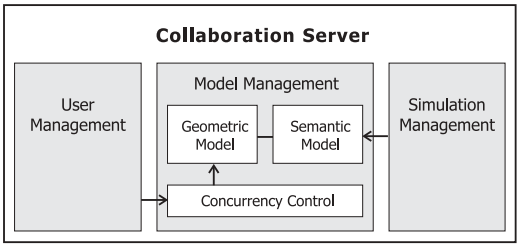

Figure 5: The modules of the collaboration server

Modifications like adding, removing or transforming obstacles are communicated from the performing client to the collaboration server. In order to avoid conflicts among the participants, the collaborative work is coordinated by means of locks. If an object is locked by a certain user it cannot be modified by any other user until the lock has been released. The collaboration server provides an event service which the clients can access in order to obtain details of any modifications.
The clients serve as the visualization and interaction interface for the engineers taking part in the collaborative session. They can be run in single-window mode capable of stereoscopic rendering for use in virtual reality environments, or in multi-window mode for use on desktop computers.
The major task of a simulation server is to link the distributed collaborative system with a particular simulation kernel. Like the clients, the simulation server listens in to the events broadcast by the collaboration server. Accordingly, it is notified of any user interaction and can forward this information to the simulation kernel. The simulation server provides an interface to start and stop the simulation and to pass on steering parameters. It notifies all clients when the simulation is started or stopped.

4 Conclusion

We have presented an interactive CFD application (Fig. 6), enabling several users to collaborate in interactive simulation and visualize its resultant data. In both versions (collaborative and non-collaborative), the visualization can be done in a virtual reality environment or on common desktop screens. Because of the fast simulation kernel and efficient communication model, the interactive simulation reacts instantly to user interaction such as modifying the geometry within a simulation domain. In future improvements, a thermal and turbulence model will be integrated into the simulation kernel as proposed in [9].

Acknowledgements

We wish to express our gratitude and thanks to KONWIHR (Competence Network for Technical, Scientific High-performance Computing) and Siemens AG for funding this research work.

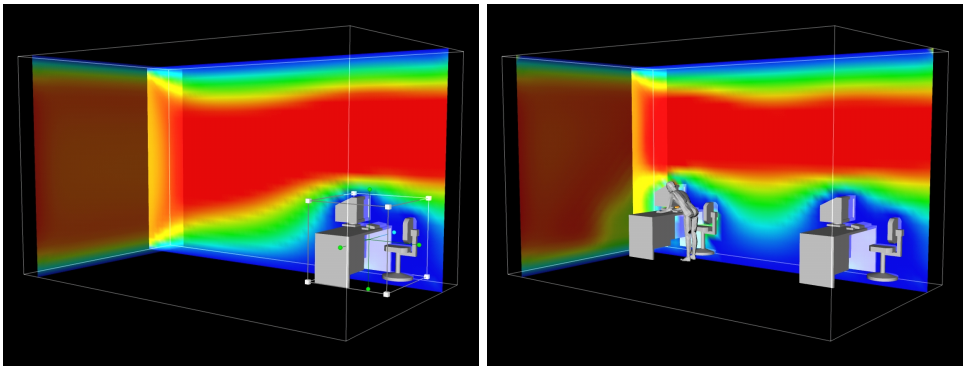

Figure 6: Snapshots of air-flow configurations for two different furniture scenarios of an office room in an interactive case study.

References

[1] Mulder, J. D., Wijk, J. van, Liere, R. van: A Survey of Computational Steering Environments, Future generation computer systems, 15(2), (1999)

[2] Succi, S.: The Lattice Boltzmann Equation for Fluid Dynamics and Beyond, Oxford University Press, ISBN 0 19 850398 9, (2001)

[3] Wenisch, P. and Wenisch, O.: Fast octree-based Voxelisation of 3D Boundary Representation-Objects, Technical Report, LS Bauinformatik, TU München, Germany (2004)

[4] Wenisch, P., Wenisch, O., and Rank, E.: Optimizing an Interactive CFD Simulation on a Supercomputer for Computational Steering in a Virtual Reality Environment, in High Performance Computing in Science and Engineering, Garching 2004, Springer, ISBN 3 540 26145 1, (2005)

[5] Wenisch, P., Wenisch, O., and Rank, E.: Harnessing High-Performance Computers for Computational Steering, in Lecture Notes in Computer Science, Volume 3666, at press.

[6] http://www3.niu.edu/mpi

[7] http://www.hlrs.de/organization/pds/projects/pacx-mpi

[8] Borrmann, A., Wenisch, P., Treeck, C. v., Rank, E.: Collaborative HVAC design using interactive fluid simulations: A geometry-focused collaboration platform. In: Proc. of the 12th International Conference on Concurrent Engineering, Fort Worth (2005)

[9] Treeck, C. v.: Gebäudemodell-basierte Simulation von Raumluftströmungen, PhD Thesis, LS Bauinformatik, TU München (2004)