

Interactive Fluid Simulations: Computational Steering on Supercomputers

Petra Wenisch

Technische Universität München, Lehrstuhl für Bauinformatik, wenisch@bv.tum.de

Abstract

This study presents a computational steering application coupling a supercomputer with a visualization and steering interface so the user can interact/control the running computational fluid dynamics (CFD) simulation. The underlying Lattice Boltzmann-based CFD computation and also the grid generation is performed on the supercomputer while steering and data visualization of the simulation results is done through a graphical front-end application on an external system. The interaction during a simulation run comprises not only the variation of parameters, but also the modification of the geometry and hence the computational grid.

1 Introduction

Nowadays, numerical simulations in the area of fluid mechanics are an important supplement to classical wind tunnel experiments in engineering practice. These simulations are typically conducted as batch processes consisting of a preprocessing step to map CAD data onto a computational grid and to define boundary conditions, followed by the computation and the postprocessing step for analysing the simulation results. Simulating a given fluid flow scenario with high resolution is still a time consuming process. To facilitate a fast preliminary and qualitative estimate of the flow field by integrating the three above-mentioned steps a computational steering application is desired.

The interactive CFD application in the present study has been developed with this kind of issues in mind. It concentrates on indoor air flow simulations and allows the user to interact with flow parameters, boundary conditions, and geometry within the simulated scene during a simulation run. We will discuss aspects regarding solver, grid generation and communication between supercomputer and steering/visualization terminal. The main focus of this paper will be put on performance investigations of the interactive application with regard to usability (responsivity) and identification of factors limiting performance.

2 Computational Steering

For the wide variety and explorative environment needed in case studies, a user would like to interact with a running simulation and to visualize the corresponding physical reaction immediately. This is the basic idea behind computational steering [1]. It requires the integration of the above-mentioned steps, viz. pre-processing, computation and post-processing, into a single environment. To fulfill the requirement of an immediate (or at least low latency) response to user interactions during a running simulation, a fast CFD solver must be run on a supercomputer or cluster and has to be coupled to a steering and visualization workstation by an efficient communication concept. In addition, the computation kernel has to allow for a fast grid modification during the simulation, and interfaces for steering and visualization should be available to the user within a single environment.

2.1 The Lattice Boltzmann Method

The Lattice-Boltzmann method (LBM) has emerged as a complementary technique for the computation of fluid flow phenomena. Common numerical methods for the simulation of fluid dynamics are based on a discretization of the Navier-Stokes nonlinear partial differential equations. The LBM represents an alternative approach consisting of a discrete microscopic model described by means of statistical physics [2]. By computing the dynamics of particle densities for a discrete number of velocities and directions at each grid point appropriately, quantities such as mass and momentum are conserved to fulfill the hydrodynamic laws.

In each time step the Lattice-Boltzmann algorithm first computes the collision of microscopic, virtual particles and updates the velocity distribution functions of these particles. Luckily, the corresponding evaluation of the new velocity distribution functions does not require data exchange with adjacent grid nodes in the spatially partitioned problem of a parallelized computation. At the end of each time step the distribution functions at each grid node are migrated to neighboring cells in the 'propagation' step. In a typical implementation the LBM is computed on a uniform Cartesian grid making this approach to solving the Navier-Stokes equation particularly well-suited for parallelization.

2.2 Grid Generation

Of course, a fast grid generation is also an essential requirement for an interactively steerable CFD application to allow the modification of the geometry, that is, e.g., inserting or deleting obstacles. With the present application the user can load arbitrary CAD-generated geometries which the grid generator (described in detail in [3]) transforms into a uniform grid representation required by the Lattice-Boltzmann method (see above). The corresponding voxelization algorithm for an optimized grid generation is based on the hierarchical space partitioning via an octree.

2.3 Communication Concept

In the following, we discuss how the communication between steering terminal and computation nodes has to be designed to meet the special technical challenges of computational steering. The visualization workstation provides the functionality to display the current data and to interact with the simulated scene at the same time. When a user interaction has occurred, the corresponding modifications are forwarded to the supercomputer. There, the new configuration is incorporated immediately into the new computation step and as soon as new results are available, they are sent to the visualization client, where the user can observe the adaptation of the fluid flow. Figure 1 shows the different components of the application and the corresponding data flows.

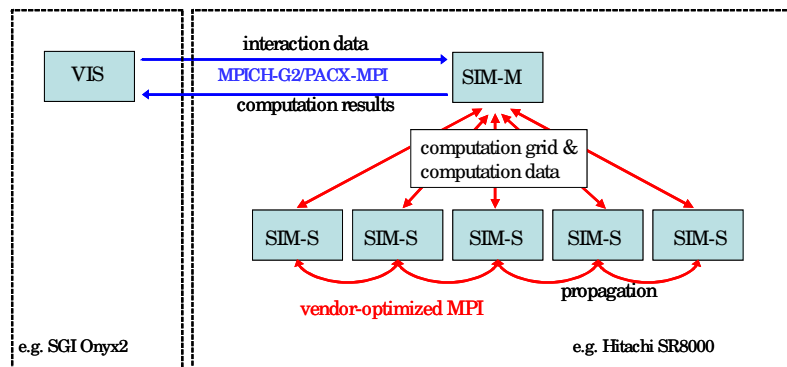


Figure 1: Application scheme showing the visualization (VIS) and the simulation side: The latter consists of a master node (SIM-M) and computation slaves (SIM-S). The VIS process is usually run on a graphics workstation, whereas the simulation is performed on a supercomputer. We have been using the application on a Hitachi SR8000 at the Leibniz Computing Center (LRZ) in Munich and an SGI Altix 3700 at Sara Computing Center, Amsterdam.

On the visualization side (VIS), an additional communication thread is employed to continually check for incoming results and to be able to forward user modifications without interrupting steering, visualization and post-processing. To keep the data transfers as short and infrequent as possible, only modifications to the setup are forwarded. Therefore, the transmission process is not triggered until after the user has completed his modifications. Since the results are also not necessarily sent at regular intervals, the receipt of data is, in essence, an event-driven process in both directions.

The slave processes (SIM-S) on the supercomputer mainly perform the LB computation. As long as no interaction has occurred, they send current results at user-defined, regular intervals and check for updated computational

grids. In the event of an interaction, a new grid is received by the simulation kernel and new results can be sent after just a few time-steps to give the user fast initial feedback depending on his manipulations. Consequently, the transmission intervals are not necessarily regular throughout the run.

To uncouple communication due to interactive steering and computation, an additional process (SIM-M) has been introduced on the supercomputer which communicates with the steering terminal. We will refer to this as the master node. It perpetually checks in both directions whether new data is available from either the slave processes or the visualization, and pre-processes the data before forwarding it to its final destination. The master also collects results of a particular time-step from all computation slaves and sends them, combined into a single message, to the visualization client to avoid the additional latencies that would arise if these messages would be sent separately. This is especially important when the network connection between supercomputer and visualization client is filtered by routers and firewalls or is otherwise limited by slow connections — maybe even due to competition with other users.

Finally, Fig. 2 reveals the main advantage of introducing this collector node: During the time-consuming transfer of results (from SIM-M to VIS), the slave processes are able to overlap computation with communication as long as the computation time is longer than the time of communication. If communication takes too long, performance saturates as can be seen for intervals of 40 or less. It is important to point out, however, that using non-blocking MPI communication on the SR8000 (and this is also true for several other MPI implementations, [4]) does not allow this overlap. Figure 2 demonstrates that without this overlap due to the missing collector node the performance of the application decreases dramatically.

Consequently, the transfer of results from the supercomputer to an external visualization and steering environment may turn out to be the limiting bottleneck. To counter this potential bottleneck, only essential data should be transferred, preferably in compressed form. Although Hitachi's SR8000 at LRZ offers a Gigabit network, no more than 230 MBits/sec were available for outgoing connections. Therefore, the application was transferred to Sara's SGI Altix 3700 in Amsterdam, to benchmark the computational steering application on different hardware. To be able to compare the measurements offline performance (i.e. visualization node also running on the supercomputer) was measured on both systems (Figure 3).

For the setup at Sara, an external visualization client was connected to the Altix machine also via Gigabit Ethernet with synthetically benchmarked 720 Mbits/sec. Comparing the saturation performances on the SR8000 and the Altix 3700 showed a performance gain of 310% as compared to 170% (see Figure 4). This corresponds to data updates every 13 seconds on Hitachi's SR8000 and only 4 seconds on the SGI Altix for 380x355x122 grid points of the example office room with dimensions 7.6m x 7.1m x 2.24m. The graphs in Figure 4 also show that even with a higher performance of the computational kernel data updates every 13 seconds cannot be surpassed with the network connection provided in the setup of SR 8000 with external visualization. Only the superior efficiency of the network at Sara is able to further decrease the data update time significantly.

3 Interactive Blood Flow Simulation

A special field of expertise of the host institute for Computational Science at the University of Amsterdam is the research on computational haemodynamics as needed for vascular reconstruction simulations. Vascular reconstruction includes surgical operations like adding shunts, bypasses and placing stents (in the case of aneurysm) or applying thrombolysis techniques, balloon angioplasty, bypasses, etc. for a stenosis. To find the best treatment is far from trivial and a simulation tool to support the verification of the operation plan may serve as a good supplement to classical approaches.

To evaluate the flexibility of the computational steering framework it was applied to a simplified artery simulation, where the user is able to (partially) block an artery or add bypasses during simulation runtime. The Lattice Boltzmann kernel used did not incorporate details of haemodynamics, but was able to get qualitative correct flow behavior ([5]). After only little adaptations interactive simulation was already possible for an adequate grid resolution using the SGI Altix 3700 and a laptop (Intel 2.13 GHz processor with ATI Mobility FireGL V5000) for visualization.

4 Conclusion

We have presented a computational steering environment for CFD and verified the communication performance regarding its applicability. While the CFD kernel is being processed continuously in the background, the simulated scene can be modified interactively. Due to a fast grid generation, even the modification of complex geometries is well supported. Performing the simulation on the SGI Altix 3700 and the visualization running on an external graphics workstation the achieved data update rate for moderately fine grids is quite satisfactory.

This study also demonstrated that the underlying computational framework is flexible enough to being adapted to a different kind of CFD application.

5 Acknowledgments

This project has received financial support from HPC-EUROPA (a project of the European Community: FP6 "Structuring the European Research Area") and KONWIHR (funded by the state of Bavaria), which is gratefully acknowledged.

The author would like to express her gratitude to Willem Vermin, Laura Leistikov and Huub Stoffers (all Sara Computing Center, Amsterdam) for their great support during benchmarking the application on the SGI Altix 3700 system. Also to Alfons Hoekstra, Lilith Abraham (University Amsterdam) and Jos Derksen (Technical University Delft) for many valuable discussions while implementing the Lattice Boltzmann turbulence model.

I also would like to thank Ernst Rank (Technical University Munich), Ulrich Rude (University Erlangen) and Peter Sloot (University Amsterdam) who

supported me already during the run-up to my HPC-Europe visit in Amsterdam.

Finally, I gratefully acknowledge the assistance by Oliver Wenisch, Irene Geiseler, Helmut Heller and Reinhold Bader (all Leibniz Computing Center, Munich) during the benchmarks on the Hitachi SR8000.

References

- [1] Mulder, J. D., van Wijk, J. J. and van Liere, R.: A survey of computational steering environments. *Future Gener. Comput. Syst.*, 15(1) (1999)
- [2] Gitter-Boltzmann-Methoden: Von der Theorie zur Anwendung, Professional dissertation, LS Bauinformatik, TU München (2001)
- [3] Wenisch, P. and Wenisch, O.: Fast octree-based Voxelization of 3D Boundary Representation-objects, Technical Report, LS Bauinformatik, TU Munchen (2004)
- [4] White III, J. B. and Bova, S. W.: Wheres the Overlap? An Analysis of Popular MPI Implementations, MPIDC99, (1999)
- [5] Abrahamyan, L.; Schaap, J.A.; Hoekstra, A.G.; Shamonin, D.P.; Box, F.M.A; van der Geest, R.J.; Reiber, J.H.C. and Sloot, P.M.A.: A Problem Solving Environment for Image-Based Computational Hemodynamics, in V.S. Sunderam; G.D. van Albada; P.M.A. Sloot and J.J. Dongarra, editors, *Computational Science - ICCS 2005: 5th International Conference, Atlanta, GA, USA, Proceedings, Part I*, in series *Lecture Notes in Computer Science*, vol. 3514, pp. 287-294. Springer, Berlin, Heidelberg, (2005).

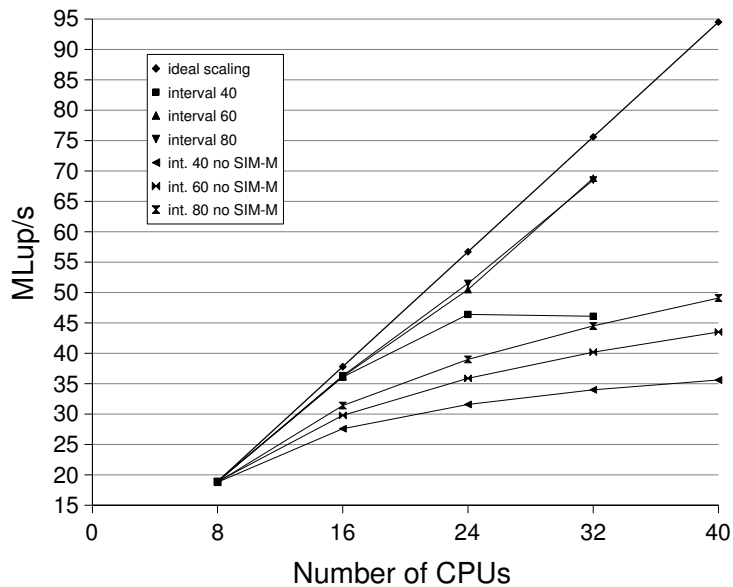


Figure 2: This graph shows the scaling behavior for various update intervals in dependence on the number of CPUs used on the Hitachi SR8000. Performance is measured in million lattice site updates per second (MLup/s) available at the visualization process. Runs with update intervals of 60 or more time steps show good scaling already. Shorter intervals, however, have a negative influence. It is evident that the application shows only poor scaling efficiency (<50%) in all cases without the master node (SIM-M). The same effect has also been observed on Sara's Altix system.

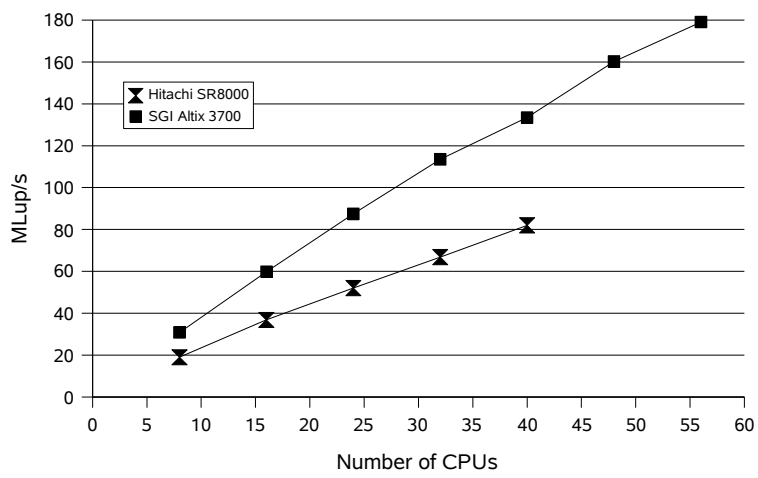


Figure 3: Offline Performance of both systems, Hitachi SR8000 and SGI Altix 3700, measured in MLup/s at the visualization process. Using Altix CPUs achieves a performance increase by a factor of 1.7 as compared to the SR8000.

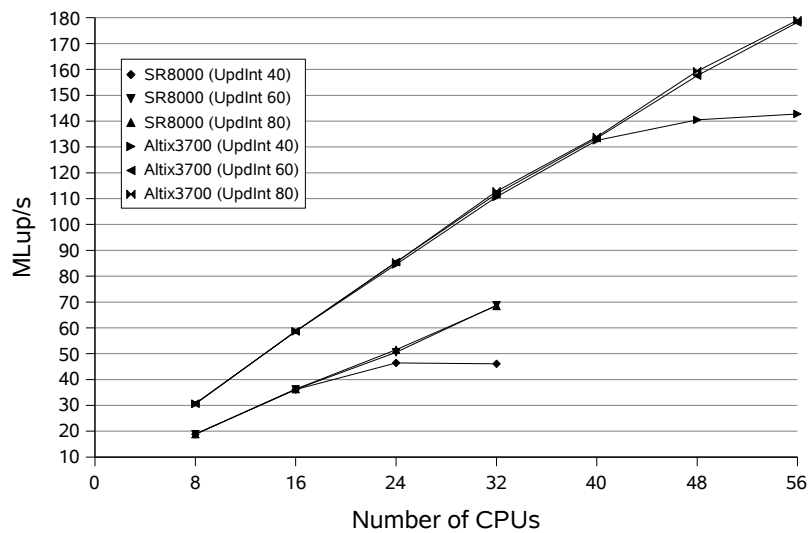


Figure 4: These pairs of graphs show the performance gain in dependence on the number of time steps between result updates to the visualization for runs on the SR8000 and Altix 3700, respectively. Both machines show performance saturation when frequently updating results, because communication time then exceeds computation time. Updating less often than every 40 time steps quickly restores good scaling behavior. It is evident that the SGI can make much better use of its outgoing Gigabit ethernet network than the older Hitachi system.