Demonstration of Supercomputing Activities

Alfred Geiger

During the inauguration of HLRS a broad variety of supercomputing-activities was demonstrated. As HLRS is involved in the development of Software for supercomputers as well as in their usage, the demos concentrated on the development of novel simulation-codes in the field of Computational Fluid Dynamics, the software-tools which are necessary for the developper and on the presentation of typical applications from engineering in the VR-lab.

Collaborative Supercomputer Simulation Visualization

Andreas Wierse

In the visualization lab, the COVISE software environment was presented, which enables researchers and engineers to directy and interactively visualize results of ongoing supercomputer-simulations. The first application shown was a vortex-breakdown simulation running on the NEC SX-4. The result of this simulation is immediately sent via the network to a graphics workstation connected to a virtual environment based on a large projection screen which allows 3D display with stereo glasses to a group of people. The second application (picture) showed the flow-visualization of an internal combustion engine. The data were generated with the STAR-HPC Code from Computational Dynamics in the framework of an EC-funded project of RUS, Computational Dynamics, Daimler-Benz, Renault and Cray Research.

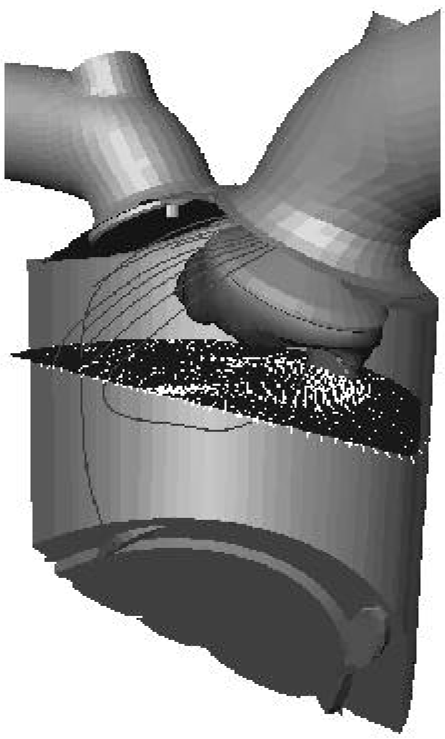


Figure 1: Flow-visualization of an internal combustion engine

Integrated Grid Generation and Flow Simulation on Arbitrary Unstructured Grids

Clemens Helf/Thomas Schmidt/Katina Warendorf/Uwe Küster

For a variety of engineering disciplines Computational Fluid Dynamics (CFD) has emerged as a powerful and important analysis tool. CFD is concerned with the numerical simulation and visualization of fluid flows using very powerful computers.

While CFD became essential in more traditional areas of application such as aircraft, turbomachinery problems and car design, other disciplines increasingly make use of this technology as it becomes available, like weather prediction, environmental sciences (air/ground water pollution models, climate models), civil engineering (room/house climatization) or medical research (blood flows). With the introduction of CFD to new disciplines, increasingly complex computational domains need to be handled. The generation of appropriate spatial discretizations (grids) for these domains currently is a limiting factor, because it is a very time consuming task in terms of human effort and gets increasingly difficult with complex domains. Furthermore, the accuracy of a computed flow solution heavily depends on the interaction of the flow solvers with the provided grid.

More flexible grid representations, self-adaptive grid generation and the integration of flow simulation and grid generation into a single process provide a way to tackle these difficulties. As an example, a simple, inviscid flow around an airfoil profile (2D) is presented.

The flow solver ceq is based on a very flexible grid representation, which is capable of supporting a direct CAD interface. The process of flow simulation starts with the single control volume grid imported from CAD. No initial grid generation is necessary to start the simulation process (figure a). Obviously, an accurate solution will not be obtained on a single control volume grid. Therefore, a heuristic refinement strategy is used to create a reasonable coarse grid, by repeated refinement of control volumes at the body surface (figure b).

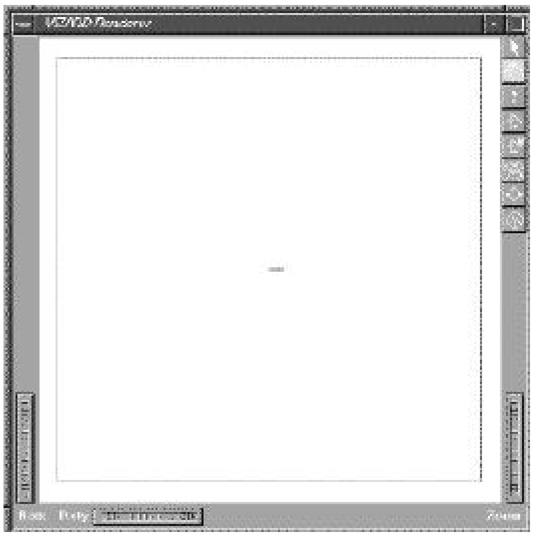
The flow solver ceq integrates grid generation into the process of flow simulation. Starting from

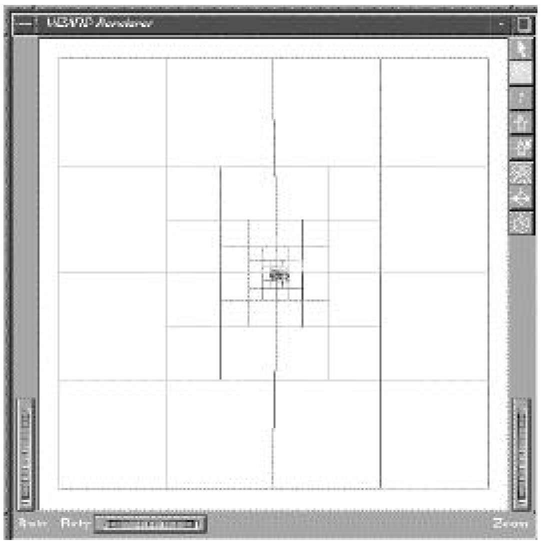
a coarse, but reasonable grid, grid and flow solution are improved together by self-adaptive refinement and recoarsening. Therefore, no extra effort is necessary for grid generation, i.e. grid generation is fully automated. Finally, a fully developed grid and an accurate solution are obtained (figure c-f).

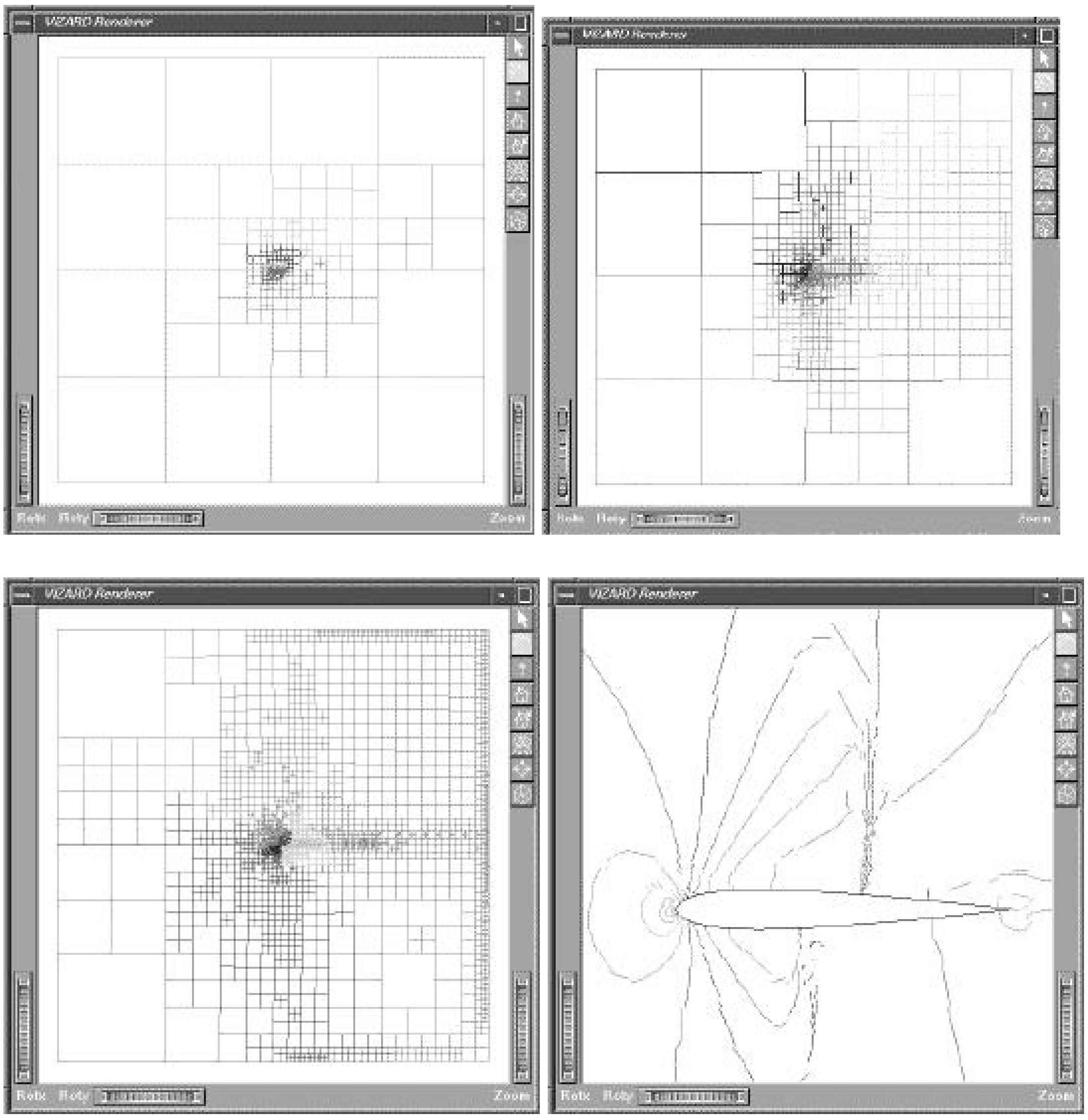
In order to provide simulation results quickly, ceq is available on vector and distributed memory parallel supercomputers. The flow solver is completely parallelized using the Distributed Dynamic Data (DDD) library. DDD automatically builds up processor interface descriptions and keeps them uptodate across transfer operations. Data at processor interfaces are synchronized by explicit interface communications. Dynamic load balancing is achieved using the Metis library. Use of parallel supercomputers enables us to solve very large problems and speeds up the simulation process. The different shadings (figure b-e) show the distribution of the computation grid to eight processors.

The vizard vizualization tool was used to create the figures presented here. The vizard directly supports complex control volumes and piecewise continuous solution representations that are used by the flow solver and user objects, independent of the flow representation. Therefore, no data transformations are necessary in order to visualize computed results. This is an essential feature for algorithmic developments and debugging. The most apparent feature of this support are discontinuous isolines as presented in figure f.

Ceq uses a very flexible grid representation, based on arbitrary polygonal control volumes, which even may be multiply connected. Therefore, no initial grid generation is necessary. Furthermore, grid generation and flow simulation are completely integrated. Therefore, no seperate grid generation is necessary and the full information from the flow solver is available for self-adaptive refinement. This provides us with a strategy for automatic grid generation. The vizard vizualization tool provides graphical support for flow analysis and code development on high-end graphics workstations.







The figure a shows the integral grid, the airfoil with a box around it.
b up to e show an adaptive grid refinement process.
In figure f the isolines ofthe machnumber around the airfoil are presented

Software Development for Supercomputers

Isabel Loebich/Jürgen Lepper

In this demonstration one step of the complete process of software development for parallel computers was shown. Based on a parallel application that is already running and produces the correct numerical and physical results, only the optimisation step was presented.

As an example a code for the simulation of turbulent flow and combustion processes in coal-fired full-scale utility boilers has been chosen. This code is being developed at IVD, University of Stuttgart, and is used to predict several physical phenomena, such as the heat transfer at the walls or pollutant emissions, e.g. NO_x. These results help to improve the combustion efficiency and to fulfil environmental legislation for pulverised coal combustion.

The numerical simulations take quite long, because full-scale utility boilers are generally large and the physical phenomena are very complex. To obtain a solution for a utility boiler with a height of 100m and a cross section of more than $600m^2$ simulation times of one week on the fastest workstation are needed, but often even longer times become necessary. Using the parallel

version of the code considerably reduces these times to a couple of hours or even less, dependent on the specific case. For the optimisation of this message passing code the characteristics of computation and communication times have to be analysed.

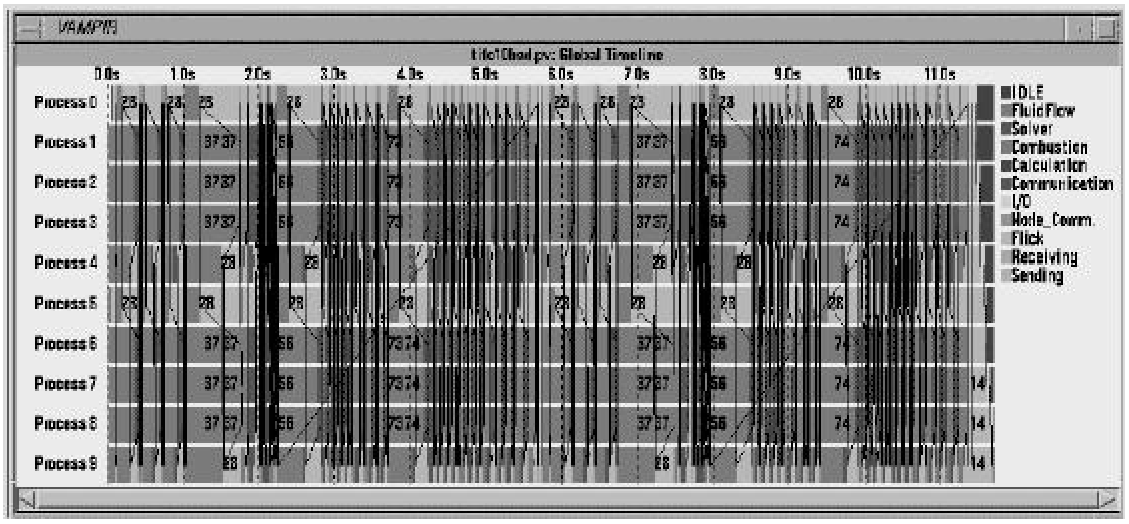
On the intel Paragon, the development platform at RUS, the runtime behaviour can be observed while the application is running with the SPV (*System Performance Visualisation*) tool, which provides an image of all processors of the Paragon. The colours of the individual processor give an impression of the actual work, and zooming provides some more information for each processor: CPU time consumption, memory usage and amount of communication. This information is gathered in pre-defined intervals from the hardware, and allows just the global overview, whether some processors are in the average less utilised than others. This is obviously not sufficient to improve the runtime behaviour and more detailed facts are needed.

With VAMPIR and ParaGraph two different tools were presented, which allow to analyse a specific run of the application on a parallel computer. For both tools data have to be collected during runtime that are used later to replay the behaviour on a workstation. To restrict these so-called `trace files' to reasonable sizes for a demonstration, two different versions of the application running on ten processors for only two iterations have been instrumented.

ParaGraph mainly allows to replay, continuously or stepwise, several aspects of the program execution. A few of the several display possibilities are shown in figure 1. One illustrates the messages that are sent and received by each processor, the animation display showing the processor activities and the communication links, and the Kiviat display which gives a clear image of the distribution of work among the processors (current utilisation and high water mark). The difference in the load balance can easily be recognized in the Kiviat display.

Figure 1: Example of the ParaGraph output

VAMPIR allows an even more detailed look at the behaviour of the application. These views show the activities of the processors over time (figure 2). The green tones represent the calculation phases, the yellow portions show phases where a processor is waiting for a message from another processor, the black lines show the messages going from one processor to another.



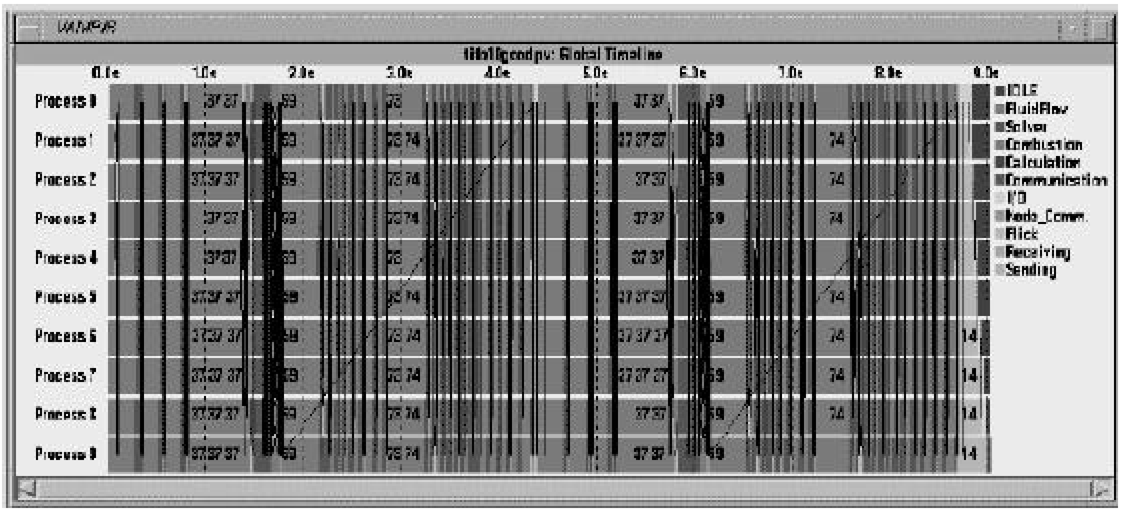


Figure 2: Example of the VAMPIR output

One can identify several components of this application: the two main modules, fluid flow (computation of the Navier-Stokes equations and the turbulence) and combustion (computation of chemical reactions and radiative heat transfer), and additionally the solvers as one major part of any numerical application. The two iterations can easily be distinguished and since they look quite similar it would be unnecessary to instrument more than that. The communication pattern reflects the implementation of the physical and numerical models.

Figure 3: Details of the VAMPIR output.

Zooming provides more detailed information, down to the individual routines of the program.

Alfred Geiger, NA-5719
E-Mail: geiger@rus.uni-stuttgart.de

Andreas Wierse, NA-5796
E-Mail: wierse@rus.uni-stuttgart.de

Clemens Helf, NA-5812
E-Mail: helf@rus.uni-stuttgart.de

Thomas Schmidt, NA-5718
E-Mail: thomas.schmidt@rus.uni-stuttgart.de

Katina Warendorf, NA-5784
E-Mail: warendorf@rus.uni-stuttgart.de

Uwe Küster, NA-5984
E-Mail: kuester@rus.uni-stuttgart.de

Isabel Loebich, NA-5991
E-Mail: loebich@rus.uni-stuttgart.de

Jürgen Lepper, NA-5801
E-Mail: lepper@rus.uni-stuttgart.de