A parallel approach for VR-based visualization of CFD data with PC clusters

A. Gerndt* T. van Reimersdahl† T. Kuhlen‡ J. Henrichs§ C. Bischof¶

Abstract

In this paper we present two CFD visualization projects based on our virtual reality software toolkit ViSTA (Virtual Reality Software University of Technology Aachen). We describe an environment for parallelized visualization algorithms to achieve a fluent visualization of large data sets. Moreover, we present a first implementation of this parallel concept on PC clusters, show attempts to resolve remaining difficulties, and introduce first results.

Key words: Visualization Algorithms, Parallel Algorithms, General Aerodynamics, Medical Applications.

AMS subject classifications: 76M27, 68W10, 76G25, 92C50.

1 Introduction

Visualization is a fundamental basis for gaining insights into the simulation results of physical processes. In particular, visualization is indispensable for the understanding of complex, dynamic processes like computational fluid dynamics (CFD), whose scientific and technical analysis requires a multi-dimensional representation of flow in space and time. The steady rise of computing performance in combination with progresses in algorithms has lead to the accomplishment of more and more complex CFD simulations. As a consequence, the requirements on visualization methods are increasing accordingly. Standard visualization tools based on animation and 2-D techniques are no longer suitable for an assessment of up-to-date CFD simulations. For this reason, efforts have been made recently to profit from Virtual Reality technology in the visualization of CFD data. In comparison to merely 2-D or at best perspective representations, VR-based, i.e. stereoscopic and user centered projection techniques, allow a much more intuitive comprehension of spatio-temporal correlations. In contrast to animation techniques, VR is inherently interactive and thus allows an interactive exploration of simulation data.

To achieve interactivity, it is necessary that a VR-based post-processor is responding to user interaction requests in real time, which is a challenging task for large data sets like they appear in complex CFD simulations with transient and/or turbulent flow situations. Typical inputs from engineers are the initiation of particle tracing in 3-D space or the definition of iso-surfaces or cut planes to be calculated and rendered. The paper we present here introduces a parallel approach with the goal to accelerate the calculation of such user requests significantly. The approach is different from other solutions [1, 2] in that it is based on a scalable cross platform concept, running on parallel "off the shelf" hardware like PC clusters instead of costly SMP architectures. The concept is being realized on the basis of ViSTA (Virtual Reality Software University of Technology Aachen) [3], which is a cross platform VR toolkit developed

*Computing Center RWTH Aachen, gerndt@rz.rwth-aachen.de, http://www.rz.rwth-aachen.de/vr
†Computing Center RWTH Aachen, reimersdahl@rz.rwth-aachen.de
‡Computing Center RWTH Aachen, kuhlen@rz.rwth-aachen.de
§Computing Center RWTH Aachen, henrichs@rz.rwth-aachen.de
¶Computing Center RWTH Aachen, bischof@rz.rwth-aachen.de

by members of the Virtual Reality Center Aachen. To cover the functionality needed for the visualization of CFD data, ViSTA makes use of the sophisticated algorithms provided by the Visualization ToolKit (VTK) [4].

In section 2 of the paper, we represent two CFD applications that are typical for research in the field of fluid dynamics at RWTH Aachen, and which demonstrate the need for VR-based visualization as well as the need for a parallelization to increase visualization performance. Section 3 describes this parallelization concept in detail and presents the results of a first test implementation, whereas section 4 draws conclusions and gives an outlook to future work.

2 CFD Applications

2.1 Spark ignition engine

Applications in the area of the visualization of computing fluid dynamics (CFD) data can be found in a wide range of use. One of our implemented VR-projects visualizes the behavior of cold gas fluids inside the cylinders of a combustion engine. This project was realized in cooperation with the aerodynamics institute at RWTH Aachen (AIA). The unsteady viscous flow problem was calculated using a finite volume solver of the AIA [5]. The calculation was executed on high performance computers of Aachen and Stuttgart, Germany.

Afterwards the result stored in 23 blocks with variable amount of grid nodes could be used for the visualization. 72 calculated time levels resulted in a total data set of approximately 600 MByte. To obtain a fluent visualization we had to interpolate further intermediate levels. 10 additional levels between each calculated time step produced a total data set for the visualization of 6 GByte.

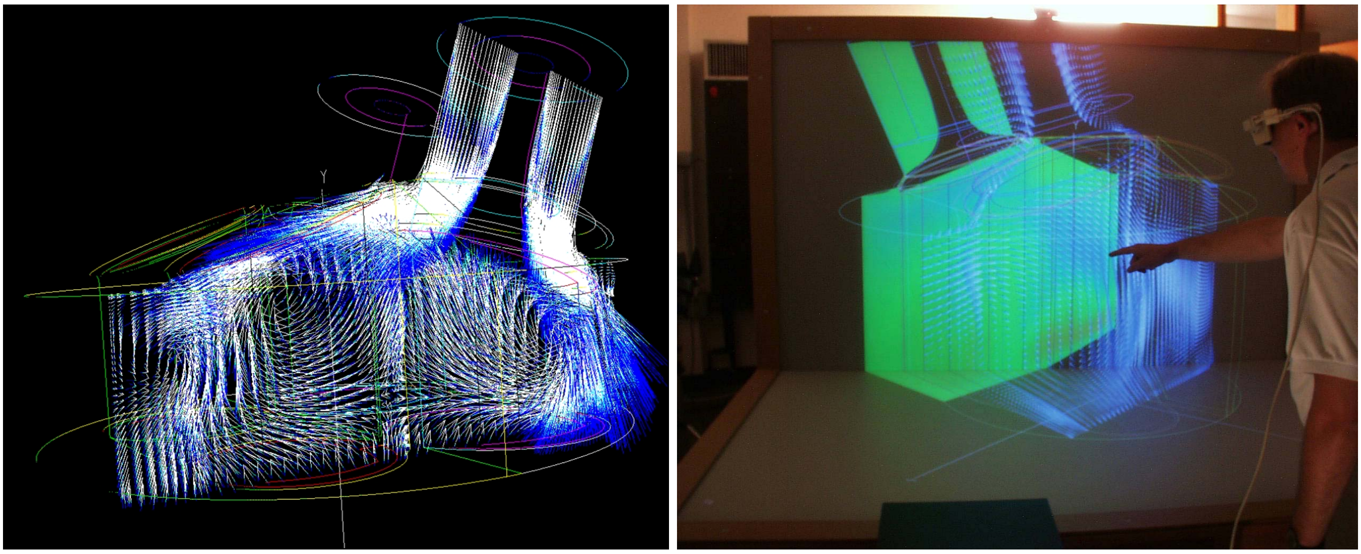

Figure 1: Vectors of velocity on cut planes (left), VR-based visualization on HoloBench (right)

The performed simulation is of great interest to the industry. The visualization should be used to analyze the efficiency of the engine. In this application, VR is a significant extension to detect important characteristics in the complex flow fields. A good efficiency can be achieved if the turbulences of the incoming air-gas-mixture are regular and hot spots can be avoided. We used particle tracing, pressure isosurfaces, color coded vertical velocity components and vectors of velocity (figure 1) on cut planes to represent the CFD data. However, to achieve a better representation of vortices, right now we are integrating advanced visualization methods like the line integral convolution (LIC) method [6, 7].

2.2 VR-based nasal surgery

One of our current projects is about the development of a virtual nose surgery tool. The human nose covers a variety of functions like warming, moistening and cleaning of breath-in air as well as the olfactory function. The conditions of

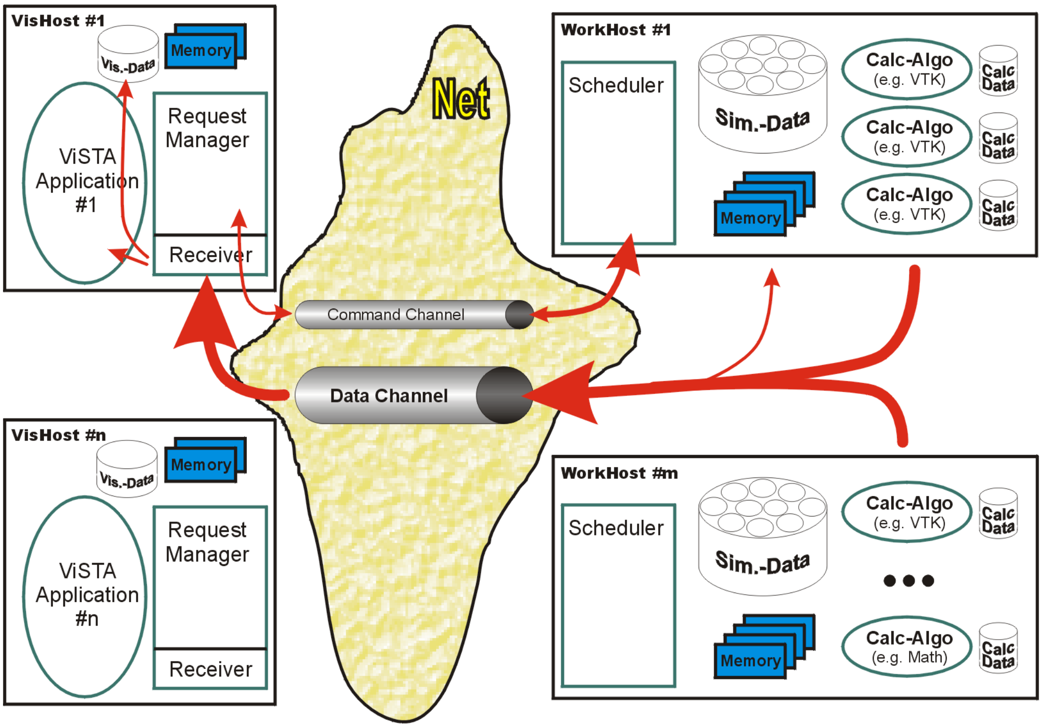

Figure 2: Important components and data flow

the flow inside the nose are essential to these functionalities, which can be restricted, for instance, by serious injury, disease or hereditary deformity. Approximately 50% of the operations to eliminate these complaints are successful. To improve this rate it is necessary to investigate the internal flow field of the nasal cavity exactly. Additionally, it is important to recognize the responsiveness of this flow field with regard to nose geometry changes.

This gives the reason to implement a computer aided surgery (CAS) system. The goal is to develop a software system that can be applied to prepare an operation and can be integrated into the education program of ear-nose-throat surgeons. For this the anatomy of nasal cavity extracted from computer tomography (CT) data will be displayed within a virtual environment. The geometry of the nose can be used to generate a grid on which the Navier-Stokes equations are solved to calculate the flow. Afterwards, an important step is to represent the pressure loss in the virtual environment as a measure of success before and during the virtual operation. The visual representation should be coupled with haptic feedback to assist the virtual operation. On completion of the virtual operation the surgeon can restart the flow simulation on the basis of the changed nose geometry. This process can be repeated iteratively until an optimum geometry is found.

3 Parallelization of visualization tasks

3.1 Design concept

In general, CFD data sets are huge data sets. The time necessary for computing the different views using different algorithms (streamlines, isosurfaces [8, 9], ...) is an important factor for user-acceptance of a post-processor for interactive exploration of the flow data. Since memory and computing power of a single machine are mostly insufficient for interactive visualization, we have decided to employ parallel computing to speed up our VR-toolkit ViSTA. The time consuming tasks are processed on dedicated parallel machines, so called work hosts, while the result is visualized on visualization hosts, where specialized graphics hardware is available.

Figure 2 shows the parallelization concept of ViSTA. On the left hand side, the visualization hosts are shown. Each of these hosts runs a copy of the same ViSTA application. As a reaction to user commands, e.g., a request to compute and display streamlines starting from specified points, a request is created within ViSTA. Each of these requests is assigned a priority, which will actually determine how fast a request is processed. This request is then passed on to a request manager, which chooses one of the work hosts — depicted on the right hand side of figure 2 — for completing the request. The request manager has to synchronize with each other, so that a single work host is not overloaded with requests while other work hosts are idle.

The request is then forwarded to the chosen work host, where it is received by a scheduler. This scheduler selects a minimum and maximum number of nodes to be utilized for the given request. These numbers depend on factors like computational speed, available memory and the capacity of the network of the parallel computer. Algorithms might actually slow down if too many nodes are used or if the network is too slow, so the number of nodes to use fulfilling a given request depends on the hardware on which the request is executed. The request is then inserted into the scheduler's work queue, which is sorted with descending priorities. As soon as a sufficient number of nodes for the request with the highest priority are available, the scheduler selects the actual nodes (up to the maximum number assigned to the request). The selected nodes then start computing, and the work hosts will transmit the result to the receiver on the visualization host that sent the request. Finally, the receiver is responsible for passing the result to the application.

This concept makes use of two different channels, as depicted in figure 2. The command channel, through which the requests are passed from the request manager to the work hosts, needs only very little bandwidth, since the messages sent along this channel are rather small (typically less than a hundred bytes). On the other hand, the results computed on the work hosts are quite large, up to several mega bytes. At this point, a network with a high bandwidth is necessary. The concept as described offers one potential optimization feature: requests might be computed in advance, preliminary to the actual user request. Even for large data sets and complex algorithms the work hosts will not be busy all the time, since the user will need some time to analyze the displayed data. During this time, requests can be computed in advance and then be cached on the visualization hosts. If a request manager receives a request, it first checks with its local cache whether this request has already been computed. If so, the result is taken from the cache and immediately displayed. For this optimization, requests with a very low priority can be generated. On the other hand, it is necessary to suspend any process working on any low priority request when an actual request from the user arrives.

3.2 Implementation

A first prototype implementation is based on the Message Passing Interface (MPI) as communication library [10]. The prototype only supports one visualization host and one work host. This allowed us to quickly code and test the described concept. A drawback of this approach is the necessity to link the MPI library to the application, even for the sequential run. For the final version of ViSTA's communication module, we consider using a specialized TCP/IP protocol for the communication between the visualization hosts and the work hosts.

For this prototype we implemented a simple parallel function for computing streamlines. The complete data set is read on each node of the work host. The computation of streamlines is then distributed equally onto the available nodes, where the result is computed independently of the other nodes. Since the visualization host expects exactly one result for each request, the computed streamlines are combined on one node and then sent to the visualization host. Work in the area of meta-computing has shown that it might actually reduce communication time when messages over a slower network are combined [11].

This simple design was feasible, since the test cases we used were small enough, so that the pre-computed data can be read completely on one node. In general, the whole raw data set is required for the calculation, which might be too large for one node. Thus, fragmentation strategies for each specific occasion have to be found (e.g., [8]). The request manager and receiver on the visualization host are implemented as separate threads that communicate via shared memory.

3.3 Demo Application

The visualization toolkit (VTK) includes a demo data set that contains the result of a CFD analysis of the air circulation inside a kitchen. It is a small structured grid with a dimension of 28 x 24 x 17 points. The visualized data set contains scalar values indicating the pressure at certain positions. The data set is provided by courtesy of Dr. L. Besse from ETH, Zurich.

Based on that demo data set we constructed an application to test the scalability and performance of our implementation of a parallelized ViSTA concept. We used the kitchen CFD data set to calculate and visualize streamlines (figure 3).

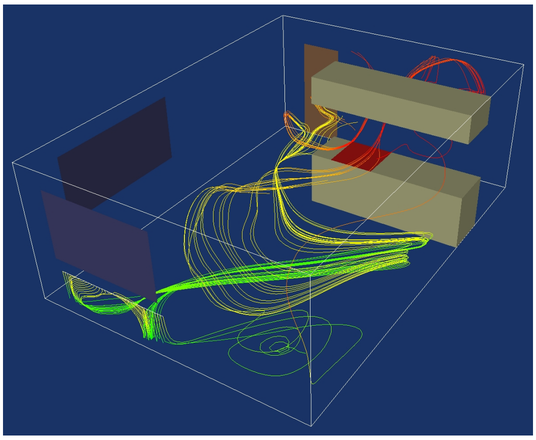

Figure 3: The kitchen demo.

3.4 Test environments

The primary goal was to separate the system that should execute the visualization and the system which is optimized for parallel computation. In our standard stand-alone VR environment we use a high performance graphics workstation of SGI, the Onyx-2 Infinite Reality 2 (4 MIPS 10000, 195 MHz, 2 GByte memory, 1 graphics pipe, 2 raster managers), which should finally be used as our visualization engine.

This system was coupled to the Siemen's hpcLine, a Linux cluster with 16 standard PC nodes (each consisting two Intel-PII processors, 400 MHz, 512 KByte level-2 cache, 512 MByte system memory), which are connected via a high performance network (SCI network, Scali Computer AS). This Linux cluster can achieve 12.8 Giga flops [12].

The parallel version was compared to sequential versions for the Onyx and hpcLine. Furthermore, we used a high performance PC graphics workstation designed by Intergraph. The Intergraph's TDZ 2000 GT1 specification is dual processor Xeon 500 MHz, 1 GByte system memory and two Intense 3D Wildcat 4000.

To determine the impact of the network bandwidth, we used different MPI implementations. On one hand we employed the native MPI implementation on the SGI and the hpcLine (ScaMPI). SGI internally uses shared memory and the hpcLine makes use of the described SCI network. Thus, these MPI implementations offer a peak bandwidth of 95 and 80 MBytes/s, respectively. But it is not possible that these native MPI libraries communicate among each another. Therefore, we used MPICH developed by the Argonne National Laboratory (ANL). MPICH is a free MPI implementation available for many platforms, in particular for each of our target architectures. Using MPICH it was possible to couple different machines and let them work together. Since MPICH does not support the SCI network, the internal bandwidth of the hpcLine was reduced to about 10 MBytes/s. This is the usual bandwidth of 100 MBits/s fast Ethernet connections also used for MPI communication between Onyx and hpcLine.

3.5 Results

Figure 4 shows the different results of the demo application when computing 1800 streamlines. Given is the time between initiating the request event at the visualization host and receiving the results at the same host. The number of nodes does not include the scheduler, which was additionally running as a separate process.

The first result is that the Linux cluster is much faster in computing the results than the SGI. This supports the claim of the so called meta-computing, where hosts of different architectures work together to solve one problem [11]. Both machines show a good speed up if they are used alone.

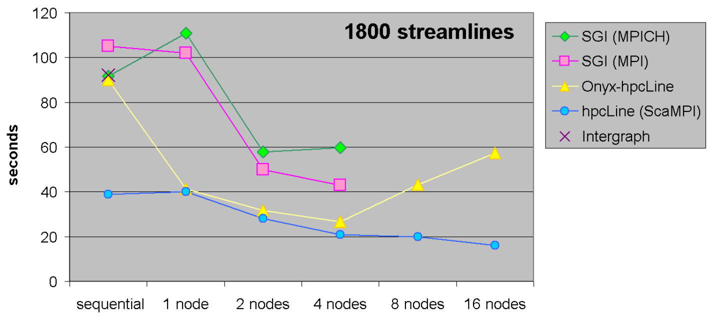

Figure 4: Comparison of run time results.

Of course, using only one node does not achieve any speed up. During the sequential run the visualization host does not contribute to the calculation of streamlines, so no additional work is done in parallel. But when more nodes are used, the calculation work is divided and the actual run time is reduced.

The influence of a fast network can be seen when comparing the result of the hpcLine alone and the coupled system SGI/hpcLine. On the coupled system, MPICH had to be used, so that only the slower fast Ethernet was used to transmit the data on the work host. As a result, a slow down could be recognized when 8 or more workers were used. This supports the importance of fast networks and the design issue that the scheduler on each work host decides on the number of nodes to use for a given request.

4 Conclusions and future work

The development of a flexible parallel environment for visualization tasks into our virtual reality software ViSTA is ongoing work, which should lead to a comprehensive VR toolkit being capable to handle complex data sets interactively. In comparison to existing VR toolkits, our efforts are towards a scalable cross platform tool running on "off the shelf" parallel hardware like PC clusters.

The general master-worker concept for parallelization we presented in this paper is in the implementation phase right now. The first simple prototype we have implemented so far shows promising run time results and emphasizes the importance of fast network components between parallel nodes. Other work in process is about the support of more platforms. For instance, we recently started tests with a MPICH version for the Win32 platform.

References

[1] D. Rantzau, K. Frank, U. Lang, D. Rainer, U. Wössner, *COVISE in the CUBE: an environment for analyzing large and complex simulation data*, in Proc. of the 2nd Workshop on Immersive Projection Technology (IPT'98), Ames, Iowa, 1998.

[2] H. Tramberend, *Avocado: a distributed Virtual Reality framework*, in Proc. of IEEE Virtual Reality 1999 Conference, Houston, Texas, 1999, pp. 14–21.

[3] T. van Reimersdahl, T. Kuhlen, A. Gerndt, J. Henrichs, C. Bischof, *ViSTA: a multimodal, platform-independent VR-Toolkit based on WTK, VTK, and MPI*, Fourth International Immersive Projection Technology Workshop (IPT 2000), Ames, Iowa, 2000.

[4] W. Schroeder, K. Martin, B. Lorensen, *The Visualization Toolkit*, Prentice Hall, New Jersey, 1998.

[5] A. Abdelfattah, *Numerische Simulation von Strömungen in 2- und 4-Ventil Motoren*, Shaker Verlag, Aachen, December 1998.

[6] B. Cabral, C. Leedom, *Imaging Vector Fields Using Line Integral Convolution*, Computer Graphics Proceedings '93, ACM SIGGRAPH, 1993.

[7] D. Stalling, *Fast texture-based Algorithms for vector field visualization*, Konrad-Zuse-Zentrum für Informationstechnik (ZIB), Preprint SC 98-40, December 1998.

[8] C.L. Bajaj, V. Pascucci, D. Thompson, X.Y. Zhang, *Parallel accelerated isocontouring for out-of-core visualization*, in Proc. of Parallel Visualization and Graphics Symposium, San Francisco, California, 1999.

[9] N. Tang, T.S. Newman, *A vector-parallel realization of the marching cubes*, Technical Report TR-UAH-CS-1996-01, UAH Computer Science Department, University of Alabama, Huntsville, 1996.

[10] W. Gropp, E. Lusk, A. Skjellum, *Using MPI - Portable Parallel Programming with the Massage-Passing Interface*, Cambridge, MIT Press, Massachusetts, 1995.

[11] J. Henrichs, *Optimizing and Load Balancing Metacomputing Applications*, in Proc. of the International Conference on Supercomputing (ICS-98), 1998, pp. 165-171.

[12] D. an Mey, *Are PC-Clusters ready for Supercomputing?*, http://www.rz.rwth-aachen.de/hpc/hpcLine, SPEC Workshop: Benchmarking Parallel and High-Performance Computing Systems, Wuppertal, Germany, 13. September 1999.