Large-Scale CFD Data Handling in a VR-based Otorhinolaryngological CAS-System using a Linux-Cluster

Andreas Gerndt, Thomas van Reimersdahl, Torsten Kuhlen, Christian Bischof
Computing and Communication Center
RWTH Aachen
Aachen, Germany

Abstract *Many of the operations to eliminate complaints concerning respiration impairments fail. In order to improve the success rate it is important to recognize the responsiveness of the flow field within the nose's cavities. Therefore we are developing a Computer Assisted Surgery (CAS) system that combines Computational Fluid Dynamics (CFD) and Virtual Reality (VR) technology. However, the primary prerequisite for VR-based applications is real-time interaction. A single graphics workstation is not capable of satisfying this condition and of simultaneously calculating flow features employing the huge CFD data set. In this paper we will present our approach of a distributed system that relieves the load on the graphics workstation and makes use of an "off-the-shelf" parallel Linux cluster in order to calculate streamlines. Moreover, we introduce first results and discuss remaining difficulties.*

Keywords: Virtual Reality, Medical Application, Computational Fluid Dynamics, Parallel Algorithms, Linux Cluster

1 The Project

The human nose covers various functions like warming, moistening, and cleaning of inhaled air as well as the olfactory function. The conditions of the flow inside the nose are essential to these functionalities, which can be impeded by serious injury, disease, hereditary deformity or similar impairments. However, rhinological operations often do not lead to satisfactory results. We expect to improve this success rate considerably by investigating the airflow in the patient's nasal cavities by means of Computational Fluid Dynamics (CFD) simulation.

Therefore we develop a VR-based rhinosurgical Computer Assisted Planning System to support the surgeon with recommendations from a set of possible operation techniques evaluated from the point of view of flow analysis [1]. For this the anatomy of nasal cavities 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, which the Navier-Stokes equations can calculate the flow simulation on. However, the CFD simulation is an enormous time-consuming task and has to be calculated as a pre-processing step. Afterwards flow features like streamlines can be extracted and visualized. During the virtual operation it is important to represent the pressure loss as a criterion of success. On completion of the virtual operation the surgeon can restart the flow simulation using the changed nasal geometry. This process can be reiterated until an optimum geometry is found.

2 The VR-Integration

A variety of commercial and academic visualization tools are available to represent flow fields, for instance as color coded cut planes. Also, streamlines or vector fields can be created. But due to the projection on a 2-dimensional display these visualization possibilities are often misinterpreted. This drawback can be avoided by integrating the com-

puter assisted planning system into a Virtual Environment. Many people can profit directly from the Virtual Reality (VR) technology. On the one hand, aerodynamic scientists can inspect boundary conditions, the grid arrangement, and the convergence of the simulation model as well as the flow result. On the other hand, ear, nose, and throat specialists can consolidate their knowledge about the flow behavior inside the nose. Furthermore, before carrying out real surgery it is possible to prepare and improve the operation within a virtual operation room.

The last aspect requires real-time interaction in the Virtual Environment with the huge time-varying data set, which resulted from the flow simulation of the total inspiration and expiration period. It is already difficult to represent the data, if the surgeon wants to explore the data set for one time level only. Head tracking and the stereoscopic projection, usually for even more than one projection plane, must not reduce the frame rate below a minimum limit. Including exploration of all time levels using additional interactive visualization techniques violates the frame-rate requirement and therefore prevents real-time interaction. In addition, if the surgeon wants to operate virtually, the planning system cannot furthermore be integrated on a usual standalone Virtual Reality system. Our Approach to handle such a complex system is a completely distributed VR system with units for the visualization and other units for the flow feature calculation and data management.

3 The Distributed VR-System

The foundation of the computer assisted planning system is the Virtual Reality toolkit ViSTA developed at the University of Technology Aachen, Germany [2]. Applications using ViSTA automatically run on different VR systems (e.g. the Holobench or the CAVE) as well as on a variety of OS platforms (e.g. IRIX, SUNOS, Win32 and Linux). ViSTA itself is based on the widely used World Toolkit (WTK). Moreover, we have implemented an interface into ViSTA in order to integrate further Open-GL-based toolkits like the Visualization Toolkit (VTK). VTK, an Open-Source project distributed by Kitware Inc., facilitates the development of scientific visualization applications [3]. Gathering all these components we could start to implement the planning system immediately without worrying about VR and CFD peculiarities.

In order to improve the performance of VR applications it may be possible to implement multi-processing functionalities. Multithreading and multi-processing are convenient features to speed up a VR application on a stand-alone visualization workstation like our multi-processor shared-memory Onyx by SGI. However, extensive calculations and huge data sets can still slow down the whole system. Therefore, we have developed a scaleable parallelization concept as an extension of ViSTA, where it is possible to use the visualization workstation for the representation of graphical primitives only. The remaining time-consuming calculation tasks are processed on dedicated parallel machines. The raw data sets, which for instance were yielded by a CFD simulation, are generally not needed on the visualization host anymore. Thus almost the whole memory and the power of all processors coupled with specialized graphics hardware are now available for the visualization and real-time interaction. In the next paragraphs a variety of additional design features are introduced in order to increase the performance even more.

3.1 The Design

Figure 1 shows the parallelization concept of ViSTA. On the left 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 at specified points, a request is created within ViSTA. Each of these requests is assigned a priority, which will actually determine how fast it is processed. Then it is passed on to a request manager, which chooses one of the work hosts, which are depicted on the right side of figure 1, for completing the request. The request manager is an internal part of ViSTA.

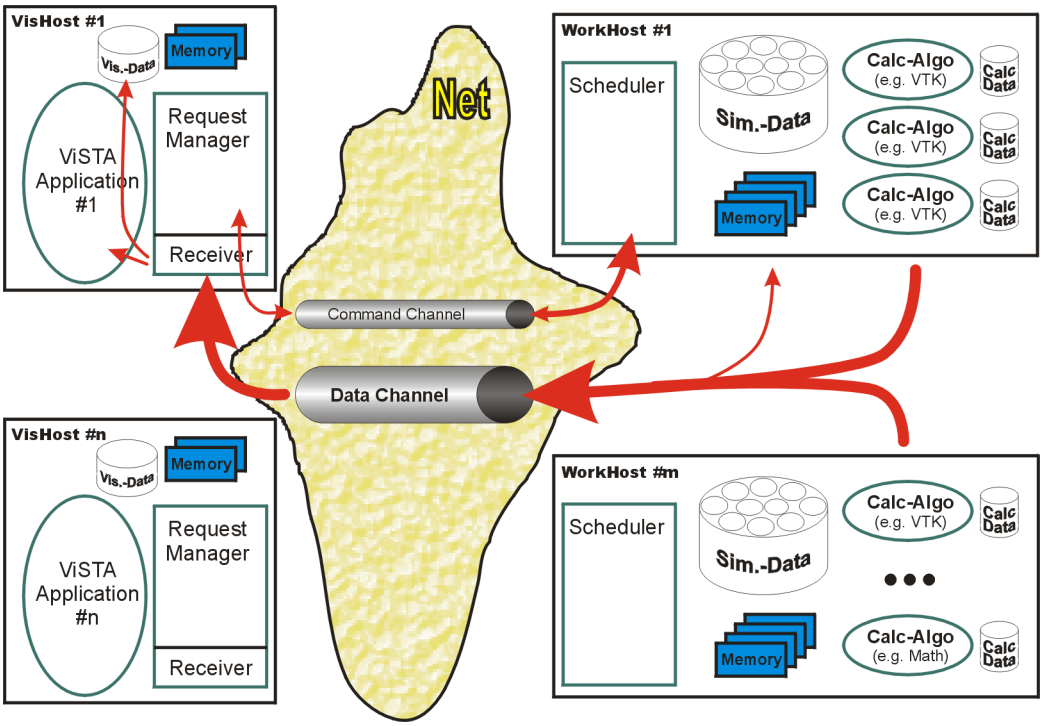

Figure 1: Important components and data flow

These request managers have to synchronize with each other to avoid a single work host being overloaded with requests, while other work hosts are idle.

Then the request is forwarded to the chosen work host, where a scheduler receives it. This scheduler selects a minimum and maximum number of nodes, which are 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 machine. 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 to fulfill a given request depends on the machine, on which the request is executed. The request is then inserted into the work queue of the scheduler, 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), which will process the request. The selected nodes then start computing, and one of the nodes will send the result to the receiver on the visualization host, which sent the request. The receiver, which is the last part of ViSTA, is responsible for passing the result to the application.

This concept makes use of two different communication channels, as depicted in figure 1. 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, which were 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 offers one potential optimization feature: requests might be computed in advance on 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 can 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 precomputation requests with a very low priority can be generated. For this optimization to work it is necessary to suspend any process working on any low priority request, when a request with a higher priority arrives.

3.2 Prototype Test Bed

A first prototype was implemented using the Message Passing Interface (MPI) as communication library [4]. This prototype supports only one visualization host and one work host. A kind of connection management dispatching the requests to systems, which are available and best-fit for this particular request, is still under construction. Thus, right now the user must determine the involved computer systems before starting the parallelized calculation task. This allowed us to quickly code and test the described concept.

MPI is speed optimized for each specific computer system. Therefore it cannot be employed for heterogeneous systems. However, our prototype implements the data receiver of the visualization host by using of the MPI technology for the communication with the work host. In general (and this is just our goal), the visualization host and the work host are different systems. Fortunately, the Argonne National Laboratory (ANL) implemented a platform independent, system crossing and free available MPI version, called MPICH. The drawback of MPICH is the loss of some of possible speed, which is understandable because MPICH is based on a smallest common communication protocol, usually TCP/IP. Therefore we compared our MPICH based prototype with the MPI versions, which only works on heterogeneous platforms. For the final version of ViSTA, we consider using TCP/IP for the communication between the visualization hosts and the work hosts, thus we can profit by the faster MPI implementations for the calculations on high performance computer.

In order to assess our prototype we merely implemented a simple parallel function for computing streamlines. The complete data set of one time level, which is to be visualized, is read on each node of the parallel work hosts. The computation of streamlines is then split up equally on 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 hosts. Work in the area of meta-computing has shown that it might actually reduce communication time when messages over a slower network are combined [5].

The simulation of the airflow within a nose is a difficult and time-consuming process. Our first nose we have examined was not a human nose scanned by a CT but an artificial nose, which was modeled as a "perfect" nose. More precisely only one cavity was modeled. Flow experiments with this model resulted in first assumptions about the flow behavior during respiration. Right now we compare our simulation results with the results of these experiments. Furthermore, the current bounding conditions and multi-block arrangements are being adapted for a converging calculation. This adjustment is ongoing work, so that we took one time step of a preliminary multi-block arrangement for our parallelized prototype [1]. However, the final multi-block solution will also profit from the parallelization concept.

The used multi-block consists of 34 connected structured blocks, each with different dimensions, which yield into a total data set of 443.329 nodes. Furthermore, for each node not only the velocity vector but also additional scalar values like density and energy are stored. Employing these informations more values, e.g. Mach number, temperature and pressure, can be determined. In order to evaluate the parallelization approach a streamline source in form of a line was defined in the entry duct of the model nose. This resulted in streamlines flowing through the whole cavity. The model nose, property distribution, and calculated streamlines are depicted on figure 2.

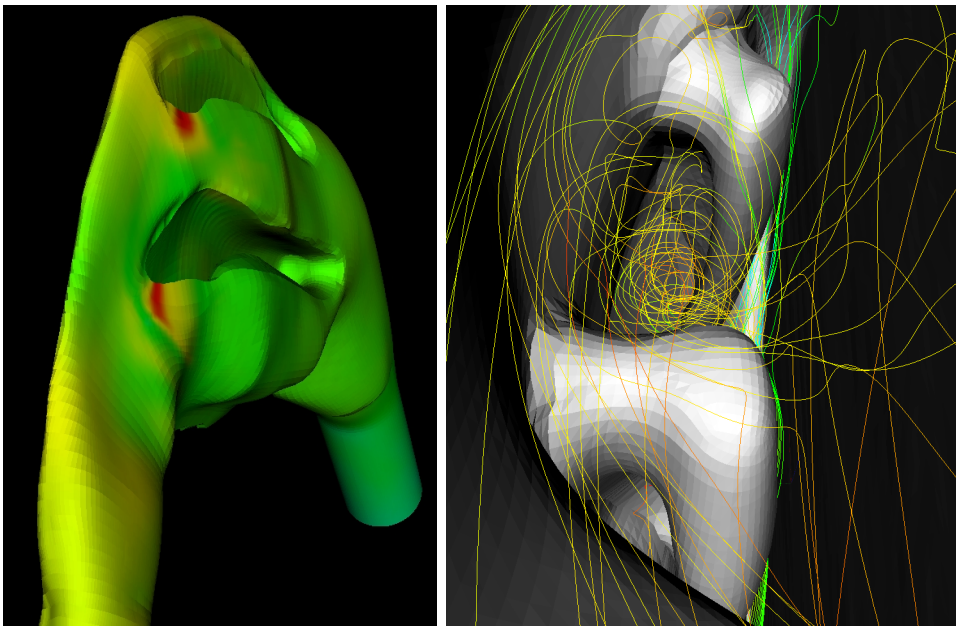

Figure 2: Outside view of one of the nose's cavities (pressure, color coded) (left), calculated streamlines inside of the nose (right)

3.3 The PC Cluster

The primary goal was to separate the system executing the visualization and the system, which is optimized for parallel computation. For our daily used 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 host for the prototype test bed.

This system was coupled to the Siemen's hpcLine at the Computing Center of the university of Aachen. The hpcLine is a Linux cluster with 16 standard PC nodes (each consists of 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 [6].

To determine the impact of the network bandwidth, we used different MPI implementations. On one hand we used the native MPI implementation on the SGI (SGI-MPI, MPI device: arrayd) and the hpcLine (ScaMPI 1.10.2, MPI device: sci), which offers a peak bandwidth of 95 and 80 MBytes/s, respectively. But, as already mentioned before, it is not possible to let these different libraries work together to couple an application on both platforms. Therefore, we used MPICH (version 1.1.2, MPI device: ch_p4), which is available for each of our target architectures, and which supports data conversion using XDR necessary for IRIX - Linux combinations. Since MPICH does not support the SCI network, the internal bandwidth of the hpcLine was reduced to about 10 MBytes/s. The Onyx and the hpcLine are connected with a 100 MBits/s Fast-Ethernet.

3.4 Results

Figure 3 shows the results of the nose application when computing 50 and 100 streamlines. The time needed for the calculation process is split up into the actual streamline calculation part, into a part needed for communication between all participating nodes, and into a part, which reorganizes the arising data structures into a unique data stream. The last step is needed, because MPI handles data streams of one data format only, e.g. only floating points, which we use for our implementation. The figure merely shows the time consump-

tion of the worker nodes. The number of worker nodes does not include the scheduler, which is additionally running but only plays a subordinary rule in this early prototype.

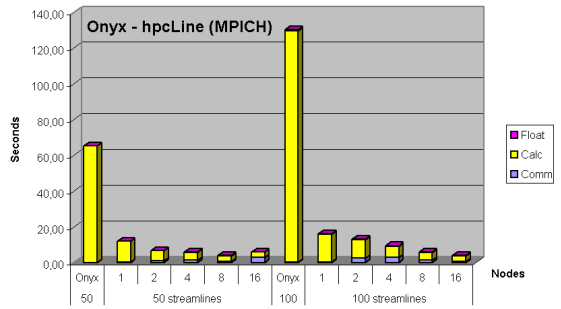

Figure 3: Calculation speed-up using different numbers of calculation nodes

As a first result the Linux cluster is considerably 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 [5]. The hpcLine shows an acceptable speed-up mainly limited by the communication overhead. The floating-point conversion does not seem to have an essential impact on calculation time. Figure 4 shows all three parts without SGI results and as separate columns. Thus they can be analyzed in more detail.

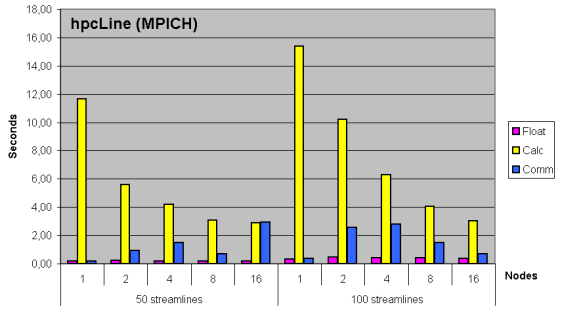

Figure 4: The behavior of the hpcLine in more detail

In contrast to earlier measurements where we used a merely simple CFD data set [7], the calculation load is not distributed equally on all node now. The distribution mainly depends on the start location of each streamline, which again controls its length and the number of line segments. As we have no means of predicting the work load, each node calculates the same number of streamlines. Therefore we could not achieve a speed-up by calculating 50 streamlines using sixteen nodes instead of eight (see figure 5). The slowest node determines the entire calculation time. In figure 5 we can recognize a calculation peek at node 4 using 8 nodes and a peek at node 7 using 16 nodes, respectively. Both peeks are nearly equal which explains the missing speed-up. Moreover, strong varying calculation loads as well as a hanging or waiting result collecting node (our node 1) increase latency times, which again influence the measured communication part.

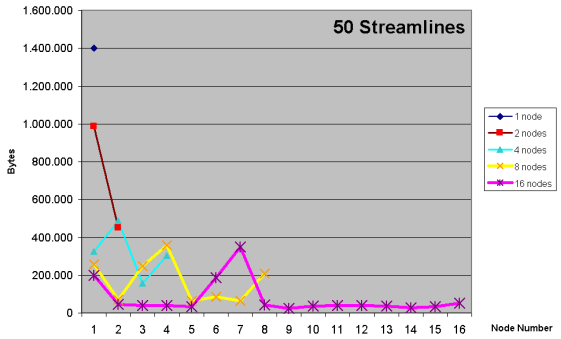

Figure 5: The resulting data size of calculated streamlines on each node

The significant 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 was employed, so that only the slower Fast-Ethernet was used to transmit the data on the work host. As a result, we finally achieve a maximum speed by utilizing of 8 nodes to calculate 50 streamlines. 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 Conclusion and Future Work

Despite the achievement of real-time interaction on the visualization host it is conspicuous that the calculation expenditure of features within the flow field should be better

balanced. This can already be achieved by a stronger integration of the scheduler, whose main job after all is the optimum distribution of incoming requests on the available nodes. This also includes the balanced distribution of one request on all nodes. Intelligent balancing strategies are going to be developed and will additionally speed-up the parallel calculation.

The simulation data was loaded in advance and was not shown in the measuring diagrams, because it is a part of the initialization of the whole VR system and therefore can be neglected. Otherwise loading the total data set into the memory of each node took approximately one minute. The Onyx has enough memory to accommodate the simulation data, however the nodes of the hpcLine already work at the limit. Larger data sets or unsteady flows definitely expect data loading on demand. Therefore, we have started developing a data management system, where only the data package containing the currently needed data block of the whole multiblock grid is loaded into memory. Leaving the current block searching the next new flow particle position forces the worker node to load the appropriate neighboring block. Memory size and topological informations control the expelling from memory. However, extensive loading and removing data from harddisc to memory and vice versa is quite expensive and should be avoided. Probably prediction approaches can make use of a set of topologically and time linked blocks. Nevertheless, should one of the structured blocks already be too large for fitting in the memory of a node a splitting strategy (half-split, fourfold-split, eightfold-split) can be applied as a preprocessing step.

The development and integration 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. However, with respect of the acquisition of an over two Tera-Flops high performance computer by SUN Microsystems that will be installed at the computing center in the next months we will also compare and adjust our development to this new architecture.

References

[1] T. van Reimersdahl., I. Hörschler, A. Gerndt, T. Kuhlen, M. Meinke, G. Schlöndorff, W. Schröder, C. Bischof. *Airflow Simulation inside a Model of the Human Nasal Cavity in a Virtual Reality based Rhinological Operation Planning System.* To appear in: Proceedings of Computer Assisted Radiology and Surgery (CARS 2001), 15th International Congress and Exhibition, Berlin, Germany, 2001.

[2] 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.

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

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

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

[6] http://www.rz.rwth-aachen.de/hpc/hpcLine

[7] A. Gerndt, T. van Reimersdahl, T. Kuhlen, J. Henrichs, C. Bischof. *A Parallel Approach for VR-based Visualization of CFD Data with PC Clusters.* 16th IMACS world congress, Lausanne, Switzerland, 2000.