Eurographics Symposium on Parallel Graphics and Visualization (2007)
Jean M. Favre, Luis Paulo dos Santos, and Dirk Reiners (Editors)

Distributed Collaborative Data Analysis with Heterogeneous Visualisation Systems

T. Düssel[1], H. Zilken[1], W. Frings[1], T. Eickermann[1], A. Gerndt[3], M. Wolter[2] and T. Kuhlen[2]

[1]Central Institute for Applied Mathematics, Research Centre Jülich GmbH, Germany
[2]Virtual Reality Group, RWTH Aachen University, Germany
[3]Louisiana Immersive Technologies Enterprise, University of Louisiana, Lafayette, LA, USA

Abstract

A system for the distributed, collaborative online visualisation in heterogeneous visualisation environments was developed and tested in the application project KoDaVis, which is part of the german optical network testbed VIOLA. The aim of KoDaVis is the visualisation of huge data sets from atmosphere research. The core of the presented distributed computer supported collaborative work system is a framework for the coupling of heterogeneous visualisation systems and the design and implementation of two distinct servers, one for the collaborative aspects and one for the direct remote access to centrally stored data. Interfaces to the VTK-based virtual reality visualisation system ViSTA and to the modular visualisation environment AVS/Express were implemented and tested. The successful coupling of these two different visualisation systems as well as the benefit of a fast optical network for parallel data access and for distributed collaboration could be demonstrated in a test setup.

Categories and Subject Descriptors (according to ACM CCS): I.3.1 [Computer Graphics]: Parallel Processing I.3.2 [Computer Graphics]: Distributed/Network Graphics I.3.7 [Computer Graphics]: Virtual Reality I.6.6 [Simulation and Modeling]: Simulation Output Analysis

1. Introduction

In 2004, the german project VIOLA [VIO] (Vertically Integrated Optical Testbed for Large Applications in DFN) started to establish a testbed for innovative optical networking. A consortium of 13 partners from industry, universities and research centres jointly work together to set up an optical network infrastructure in Germany in the area of Aachen-Cologne-Bonn and an extension to Erlangen-Nuremberg. The main goals of VIOLA are to test and evaluate advanced optical network technologies and equipment, to develop tools for dynamic bandwidth reservation and quality of service (QoS), to provide required middleware, and to analyse the behaviour of the network with a couple of network challenging sample applications. One of these applications is the KoDaVis (Collaborative Data Visualisation) project. The aim of KoDaVis is to utilise the fast optical network for the analysis of atmosphere data. This is done by the development of a heterogeneous, distributed, collaborative visualisation infrastructure, allowing direct access to huge data sets stored on a central data server.

The paper is structured as follows: In section 2 we give an overview about the area of application and a motivation for our approach. Section 3 describes the design and the three most important aspects of our system in more detail. A test set up and first experiences with the application of our system are presented in section 4. The last section discusses first results and outlines possible future improvements.

2. Background and Related Work

To understand global phenomena of the atmosphere of the earth, the dynamics and the chemistry of the stratosphere, an air layer located about 15-50 km above the earth's surface, has to be analysed. For this reason high resolution measuring instruments for the use on satellites, planes and balloons are used in big international campaigns [SMS*04]. The measured data is evaluated and compared with simulation runs, which are calculated on supercomputers [GSKK04]. Visualisation systems are used for the explorative analysis of both types of data, giving the scientists the opportunity to visualise selected features with interactive framerates. A major

© The Eurographics Association 2007.

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

problem comes from the enormous size of the data in the range of one terabyte (TB) for a single simulation run. To compare multiple simulations, computed e.g. with different parameter sets or different mathematical models, the visualisation system even has to access several TB of data. Data of this size cannot be transferred to the local workstations of the scientists as a whole. Instead it is stored on dedicated data servers or, in the case of simulation data, on the supercomputer where the calculation took place. Unfortunately, conventional visualisation systems, which are widely used in the scientific community, do not offer the possibility to get direct access to remotely stored data natively. Therefore one goal of KoDaVis is the design and implementation of a parallel infrastructure for the fast and transparent access to remote stored data directly from arbitrary visualisation systems by means of compute cluster. Although clusters could also be used for efficient, visualisation specific data processing, as implemented in the ParaView project [CGM*06], they are just used for parallel data access here.

As groups of scientists are often organised in geographically distributed teams, the development of an infrastructure for distributed collaborative data analysis is a second goal of the KoDaVis project. To allow direct communication in a collaborative session, a conventional video conference system is used. This topic is not discussed in detail in this article as KoDaVis is merely utilising existing technologies here. In addition to video conferencing, the visualisation systems have to be coupled and synchronised to allow concerted actions of the participants in a collaborative session. Visualisation systems meeting these requirements are falling into the class of the so called distributed collaborative visualisation systems [BDG*04]. Examples for such systems are COVISE [WLR93] (COllaborative VISualization Environment) and Avango (formerly named Avocado) [Tra99]. A general introduction into remote and collaborative visualisation is given in [SME02] and [FT00].

At the presentation level, immersive, projection based systems, likes workbenches [KBF*94] or CAVEs [CNSD93] can be seen as optimal tools for the visual inspection of scientific data. The so called collaborative virtual environments (CVEs) [GLG03] allow for a stereoscopic presentation of data as well as head tracking and tracked 3D interaction. However CVEs, as they are very complex and costly systems, are not available at all sites so that data analysis often is done with common desktop displays. Generally, it can be observed that distributed groups of scientist use a wide range of very different visualisation systems, both at software and at hardware level. Therefore the approach presented here has the ambition to allow collaborative data analysis within interconnections of heterogeneous visualisation systems. Our philosophy is to let each scientist work with the visualisation environment he is familiar with and which is available at his site. While the above mentioned visualisation toolkits allow collaboration only within the same base system, a framework for the collaborative coupling of different visualisation systems as well as their applications in a fast optical network is presented here.

3. System Design

The application field of the framework presented here is the distributed collaborative analysis of scientific data by geographically separated groups of scientists. Combined with a video conference system, it is aimed to serve as a platform for interactive data exploration and for discussing results. According to the classification scheme of Applegate [App91], who discriminates *where* and *when* collaboration takes place, such a system falls into the 'different place, same time' category, also known as synchronous collaboration. In synchronous collaboration, shared control data can be seen as transient. That is also true for our framework as it has not the intention to address methods for data persistence, like saving intermediate results or saving the system state. As a distributed system is designed, we have to analyse what kind of services are needed and where the service processes should be placed physically in the distributed environment. We have to distinguish between services for data fetching and services for collaboration.

As the scientific data is stored on dedicated file systems, it is a natural way to follow the common client/server approach and get access to the data by a data server located at the site where the data is placed. A well known system enabling access to remotely stored data is developed in the Open source Project for a Network Data Access Protocol (OPeNDAP) (`http://www.opendap.org`). In OPeNDAP, the access to remotely stored data is done by network-enabled interfaces to common data formats like NetCDF or HDF. A web server is used to deliver the requested data to the clients by envoking CGI scripts. Although the OPeNDAP approach is promising, it could not be used here as it does not meet some of our requirements. The main goal of our system is to gain as much profit from the fast optical VIOLA network as possible. Therefore the overhead arising from using common web servers calling CGI scripts was avoided. One important feature of our system is to exploit the bandwidth of the VIOLA network to give all clients in a collaborative session the same fast data access with a maximum of 1 GBit/sec *per client*. This is achieved by installing multiple gigabit network adapters in the data server, one reserved for each connection. The elimination of unnecessary overhead leads to the development of a purpose-built data server and an underlying slim communication protocol. We have developed a parallel data server architecture, utilising a compute cluster to drive more than one network connection simultaneously and to use the distributed main memory of the cluster as a large, fast data cache. Details of the data server are given in section 3.2.

To describe the collaborative aspects of our system, the model of Wood, Wright, and Brodlie [WWB97] is used for reference. Wood's model is an expansion of the well known

© The Eurographics Association 2007.

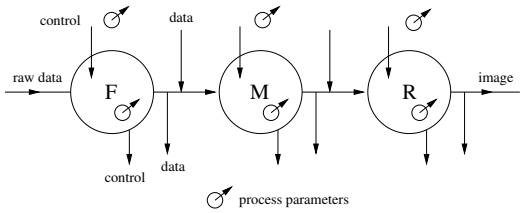

Figure 1: *Wood's model for collaborative visualisation. Collaborative visualisation is designed as a sequence of filter (F), mapper (M) and render (R) processes, enriched by im- and export of intermediate data and control parameters.*

model of Haber and McNabb [HM90], which characterises a visualisation task as a sequence of processes for data enrichment, visualisation mapping and rendering, also known as filter, mapper and render processes. Wood extended this model for collaborative systems by adding im- and export possibilities for intermediate data and control parameters (figure 1). Our system offers the possibility to distribute control parameters, which is sufficient for a synchronised distributed collaborative work environment. The synchronisation and distribution of control parameters is managed by a collaboration server, which distributes parameter changes to all clients of a collaborative session. A detailed description of the collaboration server can be found in section 3.3. The exchange of intermediate results, though possible in general, is difficult to implement here, because in our approach different visualisation systems can be connected. The distribution of intermediate data in a heterogeneous environment would result in a huge amount of conversions of complex, highly system specific data structures, which is not within the scope of the KoDaVis project.

As a distributed system is designed here, it is potentially possible to place the filter, mapper and render processes at dedicated servers. The option to relocate filter and mapper processes to a parallel computer and let the render process remain on the local visualisation hardware is discussed in section 5. An example for a complete remote rendering approach is the SGI Vizserver [Oha99], where the whole process chain takes place on a high performance SGI render server, streaming the images to visualisation clients. In our case remote rendering is not possible for two reasons. The first reason is a more technical one. Our system supports different visualisation systems with a broad range of output devices, ranging from monitors to n-sided stereoscopic projection displays. Remote generation of n stereo images for immersive, head tracked environments would result in a bandwidth of about $n * 2.1$GBit/sec for uncompressed images of size 1600x1200 pixels with 25 frames per second, even more for images with higher resolution. This very high bandwidth demand can hardly be concentrated at one output device, even in a fast optical network. The bandwidth could be reduced by using adequate compression techniques, as e.g. shown in [MC00], but due to the additional lag coming from image compression and decompression, this seems not to be a passable method for highly interactive VR-systems. Generally, we would expect that the remote image generation in conjunction with local tracking would result in an unacceptable lag, destroying the immersive impression and making fluent 3D interaction impossible. The second reason is a more conceptual one. Our system has the intention that the user, though participating in a collaborative session, can still use the visualisation system he is familiar with. So we do not want to force him to use a remote rendering software he is not experienced with.

For these reasons we decided to adapt the whole visualisation process chain of the application to specific visualisation software packages and place it on the particular local hardware. An overview of the system components and their interconnections is given in figure 2.

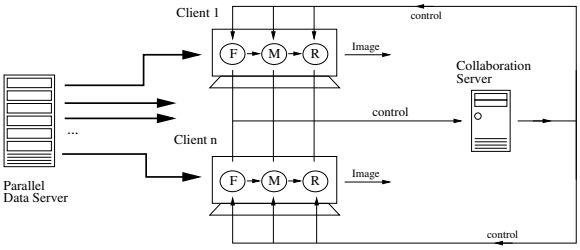

Figure 2: *The distributed KoDaVis collaboration system. All visualisation clients get direct access to remote stored data by a parallel data server. Collaboration is achieved by the exchange of control parameters, which is managed by a collaboration server.*

3.1. Support for Heterogeneous Environments

Support for visualisation systems of different kinds and flavours requires generalised interfaces both for raw data retrieval as well as collaboration. Such interfaces, available for several systems and programming languages, have to provide data conversion on at least two layers. First, one needs low-level conversions, necessary to match the machine specific representation of numbers on each supported platform. Second, one has to exchange visualisation specific objects like vectors, matrices and color tables during collaborative work. These objects need to be represented in a generic manner and converted to formats that the particular visualisation system can read and write. These visualisation specific conversions will be explained in 3.3.

In order to exchange numerical base types, arrays and strings we have chosen the communication API VISIT [FE03, EFG*05]. The read/write functions in VISIT are able to send/receive numerical data applying machine specific conversions transparently. The package comes with interfaces for C/C++, Fortran and Perl and is available under

© The Eurographics Association 2007.

an open source license from the VISIT homepage http://www.fz-juelich.de/zam/visit/.

3.2. Data Server

A typical use case in an explorative visualisation session is to visualize the dynamics of physical processes in an animation sequence. Such a sequence may consist of several hundreds of timesteps. We have tested our system with data from atmosphere research in which one timeslice amounts 3.3 MByte of raw data. In this case, the bandwith of one gigabit link makes it possible to transfer up to 25 timesteps per second which allows for interactive workflow.

In collaborative sessions, client requests for a single timestep are synchronised via the collaboration server. The data server is designed to fulfil requests of several clients concurrently and to deliver the data in a time that should be essentially independent of the number of clients. Therefore the most important feature of our data server is to provide maximal bandwidth for each connected client. This is achieved by using multiple gigabit network adapters. In order to handle the data efficiently we have designed a parallel cache architecture with strided pages for the collaborative scenario. When n cluster nodes are used, every loaded data array (e.g. the data for one property) is split into n subarrays. Each node only stores one of these parts. When a client requests the array, it is reassembled on all nodes by means of the fast internal cluster network. This mechanism is combined with a caching strategy for the sub arrays.

3.2.1. Parallelism

We have designed and implemented a parallel data server which can be started on a cluster with multiple external network interfaces. Within our design, the parallelism is exploited twofold. First, we use multiple server processes, each bound to a separate network interface, in parallel. Second, we use the internal network and the distributed memory of the cluster to manage collective retrieval and efficient caching of the archived data.

The whole parallelisation is implemented with MPI. Our data server can be started on many clusters which support MPI. Given the situation that the requested data are stored locally on each of the server nodes, no communication between these nodes is necessary to achieve parallel responsiveness. In fact the simulation data we have tested our system with, is stored on a non-parallel filesystem which is mounted separatly on each of the cluster nodes.

Reading the complete data on each of the server nodes from a non-parallel file system is inefficient for two reasons. First, the connection from the file system to the cluster becomes a bottleneck through which the data have to pass N times for N connected clients. We have designed a system, which reads the data once from the archive and transfers them to the nodes involved using the internal network

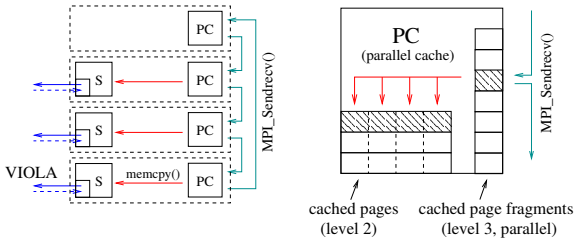

Figure 3: *On the left side, the architecture of the parallel data server is shown. The server processes (S) resp. the parallel cache processes (PC) are separate MPI tasks. On the right side, the cache hierachy is shown. Blocks of the strided data array are assembled on each server process with the MPI_Sendrecv() command. The first cache level is within the server process and not sketched here.*

of the cluster with maximal efficiency. Second, the common filesystem caching is applied on each of the server nodes separately and the available memory is not used as a whole. We have implemented a parallel cache which is linked to the server nodes via MPI. Each of the server nodes is connected to one of the cache nodes and hosted at best in the same SMP node. The number of cache nodes can exceed the number of server nodes. The architecture is sketched in fig. 3.

3.2.2. Implementation Details

We have implemented a cache using the CLOCK replacement policy [Cor68] algorithm. In order to provide a send buffer for the server processes, a reconstruction buffer for the cache processes and the parallel cache itself, the whole caching is organised in three layers.

The flow control within our parallel cache is managed by means of a token ring. If a client request is received by one of the cache nodes and if the requested array is not stored in the cache, our cache works as follows:

1. The cache process to which the request was passed acquires a cycling token which indicates, that the state of the parallel cache is *idle*. Then it reads the requested array from a file system.
2. Then it sends a token indicating a striding cycle to the next cache process in the ring. That followed by sending the $N-1$ page fragments which have to be placed in the pages of the *parallel cache* on the other nodes. After the striding cycle *each* cache node holds one fragment of the original page in its *parallel cache* (*level 3*).
3. Afterwards, a token indicating a reconstruction cycle is passed. After that, each cache node holds a copy of the complete array in its *reconstruction cache* (*level 2*). This concurrent reconstruction is due since we have to expect identical requests on the other server nodes following immediately in a collaborative session.
4. From the *reconstruction cache* the array is copied to the

© The Eurographics Association 2007.

send cache (*level 1*) via MPI. From there it is send to the Client which requested it.

The time needed for reading the array from the archive cannot be reduced within our system. The time to stride the arrays over the cache nodes as well as the time to reconstruct the arrays concurrently depends basically on the size of the array and the throughput of the internal cluster network. Since we send/receive $N-1$ fragments with size $\frac{1}{N}$ the number of nodes involved appears only on the level of latencies if the cluster provides independent bandwidth between adjacent nodes. The time needed to read, stride and/or reconstruct the arrays concurrently overlaps with the time that is needed to synchronise the requests among all the clients in a collaborative session.

The number of cache pages in each layer is adjustable to different user requirements. For a maximal number of cache pages, the *parallel cache* can be instanciated using a maximum of the available memory and the other cache levels are instanciated with only one page each and serve as simple buffers. In collaborative scenarios with delays between concurrent requests, it is useful to instanciate the *reconstruction cache* with more than one page. When users sliding forth and back in small time intervals, the *send cache* can be extended to speed up array retrieval by avoiding any MPI communication within the cluster in such cases.

3.3. Collaboration Server

To integrate multiple applications into a computer supported collaborative work (CSCW) environment means to synchronise internal state variables of the applications appropriately. Generally, there are two approaches to do this. In the *event based* approach, an application has to publish any event triggered by user interaction or by the system, which leads to a new state. Typically the events are mouse clicks or keyboard inputs. These events have to trigger analogous events in the other applications to change their state accordingly. In the *state based* approach, an application has to publish changed states. These states have to be fed into the other applications directly. Both approaches can be combined reasonably.

Another criteria is the way, how events/states are published in terms of the topology of connections. One way is the client/server approach, where a central CSCW server is set up and each client establishes a connection to it. Another way is to connect each client with each other directly. This *peer to peer* (P2P) approach can be more robust as it does not depend on a single component and therefore has no single point of failure. Also, even though it has a connectivity of N^2, such a system could be scaled to a number of clients that is sufficient for our purposes. On the other hand consistent flow control in a distributed system with P2P connectivity is hard to manage. Therefore we have chosen a design with a central collaboration server which brings all incoming messages into a unique linear sequence before publishing changed states. Our collaboration server works according to the principle of a repeater. Incoming objects from one of the clients are repeated to any client which has registered for this object previously. Sending the objects from the server to the clients requires an acknowledgement. This way the whole system becomes synchronised. The slowest of the clients defines the speed of the whole system.

Our system allows to define sharable datatypes. After log on to the server, a client can register as a sender, a receiver, or a sender and receiver for any of these types. Since this registration can be altered during a session, we have a basis for many different collaboration scenarios. Actual we have defined structures for data retrieval parameters, view parameters and visualisation options. The data retrieval parameters specify the selected array that the visualisation clients have to retrieve from the data server. The view parameters contain an object transformation matrix and the visualisation options specify e.g. the selected orthoslice and the mapping from data values to colour table entries.

4. System Validation

Two conceptually different visualisation toolkits were integrated in our framework, and a visualisation application for the given atmosphere data was implemented both of them. First, the modular visualisation environment AVS/Express [UTFK*89] was chosen, because it is widely used in the scientific community for desktop based visualisation. Second, as an example for a VR capable software package, the VTK-based toolkit ViSTA FlowLib [vRKG*00, SGvR*03] was selected. For validation purpose a simple visualisation application was implemented which renders a selected data property for one atmospherical height level on a spherical surface. The selection of timestep and data property as well as the viewing angle, scaling and color mapping parameters are synchronised in the distributed system.

The given atmospheric and environmental data consist of 50-100 arrays describing different chemical tracers or physical properties. Each of these arrays represents the global atmosphere (3D) or the surface of the earth (2D) in several hundreds of timesteps. A typical use case in an explorative visualisation session is to visualise the dynamic of physical processes in an animation sequence by cycling through all timesteps. Our system should serve 2-8 connected clients, sending the same request concurrently to the data server. These requests need to be answered simultaneously.

Our test setup was made up of one AVS/Express client and two ViSTA FlowLib clients. The AVS/Express system and one ViSTA FlowLib client were located in Sankt Augustin (near Bonn). The ViSTA FlowLib system was connected to a single sided, backprojection stereo-wall with headtracking. A further ViSTA client was installed on a render cluster at the RWTH Aachen University, attached to a CAVE-like display. Both sites were connected with the video-conference system DVTS [OKS*00] (figure 4).

© The Eurographics Association 2007.

Figure 4: *Left: Screenshot from the AVS/Express application. Middle: Video transmission from the CAVE-like system at RWTH Aachen to St. Augustin (approx. 80 km). Right: Screenshot of the Vista FlowLib application.*

The two servers were started on computers in the Research Centre Jülich. The parallel data server has been launched on a Cray-XD1 cluster and the collaboration server ran on a Linux PC. Connections between the visualisation clients and the server components have been established via the VIOLA network, using the web service ARGON [BPE*] to establish the connections between the sites and to reserve the needed bandwidth. The main components of the test setup can be seen in figure 5.

4.1. Results

The tests we have made with this setup showed that distributed data analysis is feasible even in a heterogeneous distributed environment. Though each client uses its own presentation medium, from conventional monitor up to a CAVE-like system, as well as its own software package, every participant could easily join in a valuable discussion and cooperative data exploration. Our experiences showed that equality on a per pixel basis of the rendered images on each site is not necessary. Instead it is sufficient to generate similar images, showing the same features with corresponding visualisation methods. This is valid at least under the proposition that the underlying methods are relying on the same or very similar algorithms, which is a reasonable assumption as standardised algorithms are implemented in many visualisation packages. Certainly there is also a class of critical visualisation algorithms where the final result differs highly in dependence from subtle implementation details. An example is the generation of streamlines by means of time-integration. The task of integrating those algorithms into our framework has to be examined more closely in the future.

It is also useful to study the behaviour of the system in the VIOLA network. For pure communication without visualisation, transport times of 40 ms for chunks of 3.3 MByte, which is the size of one property of the data for one timestep, could be measured, resulting in an effective bandwidth of 660 MBit/sec for each client independently. This throughput theoretically allows rates of 25 timesteps/sec, which is suitable for interactive workflow. Coupled with the visualisation clients, about 140 ms total were needed for retrieving and rendering one timestep due to additional 100 ms for data processing and rendering. This results in an update rate of about 7 timesteps/sec in a real world application. However, although we could not reach the pure network-limited rate of 25 timesteps/sec, using the VIOLA network results in a significant improvement over fast-ethernet, where the transport time of 400 ms and the rendering time of 100 ms sum up, giving a rate of only 2 timesteps/sec. Further improvements within the data-processing and rendering part of the clients will yield higher update rates.

For the collaboration of N clients connected to the collaboration server and sending/receiving short messages, the maximal number of messages is limited by the network latency. The other way round, this means for a desired update rate of u updates per second, the network latency must not be greater than $1/(2uN)$ seconds. This accounts for the acknowledgements required to guarantee a consistent flow control. The latencies in the VIOLA network are about 1 ms and thus sufficient for synchronising fast updated states, e.g. from mouse interaction, in our collaborative scenario. With four clients connected to the collaboration server we have proven that update rates of at least 30 fps are feasible. With these results we reached the range of update rates which in [FKE98] are suggested to be sufficient for tele-immersive work, that is to say 100 ms for data updates and 30 ms for control updates.

5. Discussion and Future Work

With the implementation of our framework, we were able to demonstrate that collaborative data exploration in heterogeneous visualisation environments is possible and useful, and that remote data access as well as collaboration gains profit from an advanced optical network. Though the presented system is in an early stage of development, it can be applied by the end user as is. On the other hand it could be improved in certain aspects.

It is easy to see that a major drawback and overhead comes from the need to re-implement the specific visualisation application, the so-called *logical visualisation design*, for every single base visualisation system participating in

© The Eurographics Association 2007.

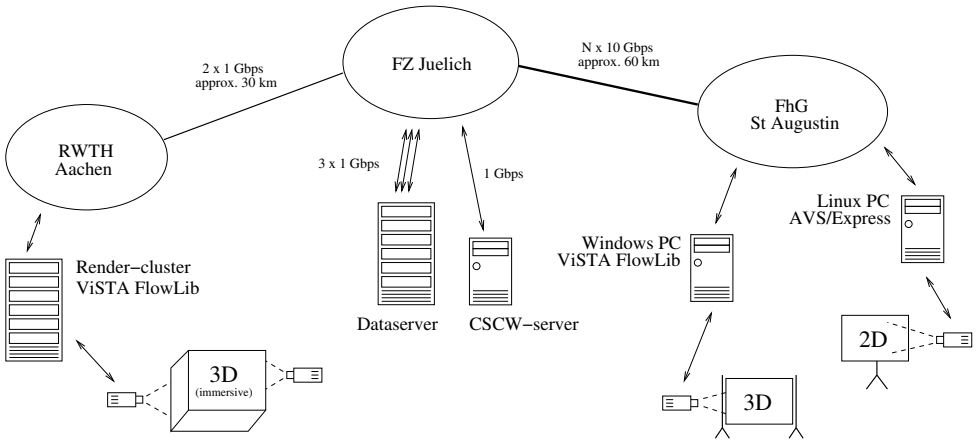

Figure 5: *The KoDaVis demonstration environment with three visualisation clients, data and collaboration server.*

a heterogeneous collaborative session. This problem can be solved by the definition of a generalised meta description of the logical design in conjunction with conversion routines, translating the meta description into actual visualisation processing networks for all used base systems on the fly. Having this method established, it would also be possible to store the logical design on a server and load it at the start of a session, as it is described in [BDG*04]. This way the user could choose from a variety of different visualisation applications, all offered by the server, at runtime.

During our tests, it turned out that the filter, mapper and render processes, which, in our approach, are computed on the local visualisation hardware, are likely to be a bottleneck and could be a limiting factor, just when processing huge sets of scientific data. To overcome this bottleneck, and already mentioned in section 3, it is possible to place time consuming filter and mapper processes on fast parallel servers. This approach is implemented in the Viracocha framework [GHW*04], developed at RWTH Aachen University, where VTK modules are used for parallel data processing of visualisation specific tasks. An integration of Viracocha with our framework could solve upcoming performance problems. In this case the local base visualisation system would be used for rendering and user interaction only. However, the use of Viracocha would be almost transparent for the end user, as the look and feel of the local system remains nearly the same with or without it.

Another important topic of research is the integration of services for collaboration, data fetching and visualisation into grid infrastructures. Enhanced by appropriate grid compatible interfaces, e. g. based on the Open Grid Service Architecture (OGSA) [FKNT02], grid enabled versions of our data and collaboration server could be implemented. In this way, the services for data fetching and collaboration presented in this paper could be made available for a wide range of user groups and applications in the grid.

6. Acknowledgements

The work reported was supported by the German Federal Ministry of Education and Research (BMBF) under grant number 01AK605L.

References

[App91] APPLEGATE L. M.: Technology support for cooperative work: A framework for studying introduction and assimilation in organizations. In *Journal Organizational Computing* (1991), pp. 11–39.

[BDG*04] BRODLIE K. W., DUCE D. A., GALLOP J. R., WALTON J. P. R. B., WOOD J. D.: Distributed and collaborative visualization. In *Computer Graphics Forum* (2004), vol. 23, pp. 223–251.

[BPE*] BARZ C., PILZ M., EICKERMANN T., KIRTCHAKOVA L., WÄLDRICH O., ZIEGLER W.: Co-allocation of compute and network resources in the viola-testbed.

[CGM*06] CEDILNIK A., GEVECI B., MORELAND K., AHRENS J., FAVRE J.: Remote large data visualization in the paraview framework. In *Eurographics Symposium on Parallel Graphics and Visualization (EGPGV) 2006* (Braga, Portugal, May 11-12, 2006), pp. 163–170.

[CNSD93] CRUZ-NEIRA C., SANDIN D. J., DEFANTI T. A.: Surround-screen projection based virtual reality: The design and implementation of the CAVE. In *Computer Graphics Proceedings, Annual Conference Series, ACM Press* (New York, 1993), pp. 135–142.

[Cor68] CORBATO F. J.: A paging experiment with the multics system. *MIT Project MAC Report MAC-M-384* (May 1968).

[EFG*05] EICKERMANN T., FRINGS W., GIBBON P., KIRTCHAKOVA L., MALLMANN D., VISSER A.: Steering UNICORE applications with VISIT. *Philosophical*

© The Eurographics Association 2007.

Transactions of the Royal Society of London Series A, 363 (2005), 1833, 1855–1865.

[FE03] FRINGS W., EICKERMANN T.: VISIT: Ein Tool zur Online-Visualisierung und Steuerung von parallelen Simulationsrechnungen. *Mitteilungen - Gesellschaft für Informatik e. V., Parallel-Algorithmen und Rechnerstrukturen, ISSN 0177-0454, 20* (2003), 35–42.

[FKE98] FOSTER I., KESSELMAN (EDS.) C.: *The Grid: Blueprint for a New Computing Infrastructure*, 1. ed. Morgan Kaufman Publishers, Inc., San Francisco, California, 1998.

[FKNT02] FOSTER I., KESSELMAN C., NICK J., TUECKE S.: The physiology of the grid: An open service architecture for distributed systems integration, January 2002. Technical Report, Argonne National Laboratory.

[FT00] FRIESEN J. A., TARMAN T. D.: Remote high-performance visualization and collaboration. *IEEE Computer Graphics and Applications 20*, 4 (2000), 45–49.

[GHW*04] GERNDT A., HENTSCHEL B., WOLTER M., KUHLEN T., BISCHOF C.: Viracocha: An Efficient Parallelization Framework for Large-Scale CFD Post-Processing in Virtual Environments. In *The International Conference for High Performance Computing and Communications* (November 2004).

[GLG03] GOEBBELS G., LALIOTI V., GÖBEL M.: Design and evaluation of team work in distributed collaborative virtual environments. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (2003), pp. 231–238.

[GSKK04] GÜNTHER G., SCHILLER C., KONOPKA P., KREBSBACH M.: Simulation of transport processes in the tropopause region during spurt using a lagrangian model. In *Geophysical Research Abstracts* (2004), vol. 6, p. 2609.

[HM90] HABER R. B., MCNABB D. A.: Visualization idioms: A conceptual model for scientific visualization systems. *Visualization In Scientific Computing, B. Shriver and G. M. Neilson and L. J. Rosenblum (eds), IEEE Computer Society Press* (1990), 74–93.

[KBF*94] KRÜGER W., BOHN C., FRÖHLICH B., SCHÜTH H., STRAUSS W., WESCHE G.: The responsive workbench: A virtual work environment. In *IEEE Computer* (1994), pp. 12–15.

[MC00] MA K.-L., CAMP D. M.: High performance visualization of time-varying volume data over a wide-area network. In *Proceedings of the 2000 ACM/IEEE conference on Supercomputing* (Dallas, Texas, United States, 2000).

[Oha99] OHAZAMA C.: White paper OpenGL vizserver, SGI, 1999.

[OKS*00] OGAWA A., KOBAYASHI K., SUGIURA K., NAKAMURA O., MURAI J.: Design and implementation of DV based video over RTP. In *Packet Video Workshop 2000* (May 2000).

[SGvR*03] SCHIRSKI M., GERNDT A., VAN REIMERSDAHL T., KUHLEN T., ADOMEIT P., LANG O., PISCHINGER S., BISCHOF C.: ViSTA FlowLib - A Framework for Interactive Visualization and Exploration of Unsteady Flows in Virtual Environments. In *Proceedings of the 7th Internationial Immersive Projection Technology Workshop and 9th Eurographics Workshop on Virtual Environment* (May 2003), ACM SIGGRAPH, pp. 77–85.

[SME02] STEGMAIER S., MAGALLÓN M., ERTL T.: A generic solution for hardware-accelerated remote visualization. In *VISSYM '02: Proceedings of the symposium on Data Visualisation 2002* (Aire-la-Ville, Switzerland, Switzerland, 2002), Eurographics Association, pp. 87–ff.

[SMS*04] STEFANUTTI L., MCKENZIE A. R., SANTACESARIA V., ADRIANI A., BALESTRI S., BORRMANN S., KHATTATOV V., MAZINGHI P., MERKULOV S., MITEV V., RUDAKOV V., SCHILLER C., TOCI G., VOLK M., YUSHKOV V., FLENTJE H., KIEMLE C., NOONE K., REDAELLI G., CARSLAW K. S., PETER T.: The ape-theseo tropical campaign: An overview. In *Journal of Atmospheric Chemistry* (2004), vol. 48, pp. 1–33.

[Tra99] TRAMBEREND H.: Avocado: A distributed virtual reality framework. In *Proceedings of the IEEE Virtual Reality* (Houston, Texas, 1999), pp. 14–21.

[UTFK*89] UPSON C., THOMAS FAULHABER J., KAMINS D., LAIDLAW D. H., SCHLEGEL D., VROOM J., GURWITZ R., VAN DAM A.: The application visualization system: A computational environment for scientific visualization. *IEEE Comput. Graph. Appl. 9*, 4 (1989), 30–42.

[VIO] VIOLA: Vertically Integrated Optical Testbed for Large Applications in DFN. http://www.viola-testbed.de/.

[vRKG*00] VAN REIMERSDAHL T., KUHLEN T., GERNDT A., HEINRICHS J., BISCHOF C.: ViSTA: a multimodal, platform-independent VR-toolkit based on WTK, VTK and MPI. In *Fourth International Immersive Projection Technology Workshop (IPT2000)* (Ames, Iowa, 2000).

[WLR93] WIERSE A., LANG U., RÜHLE R.: Architectures of distributed visualization systems and their enhancements. In *Presented at fourth Eurographics Workshop on Visualization in Scientific Computing* (Abington, U.K., 1993).

[WWB97] WOOD J. D., WRIGHT H., BRODLIE K. W.: Collaborative visualization. In *Proceedings of IEEE Vidualization '97 R. Yagel and H. Hagen (eds)* (1997), pp. 253–259.

© The Eurographics Association 2007.