FORSCHUNGSZENTRUM JÜLICH GmbH
Zentralinstitut für Angewandte Mathematik
D-52425 Jülich, Tel. (02461) 61-6402

Technical Report

Application Steering in a Collaborative Environment

John Brooke, Thomas Eickermann, Wolfgang Frings,
Paul Gibbon, Lidia Kirtchakova, Ulrich Lang**, Daniel
Mallmann, Mark McKeown*, Stephen Pickles*, Andrew
Porter*, Mark Riding*, Mathilde Romberg, Anke Visser,
Uwe Wössner***

FZJ-ZAM-IB-2004-09

August 2004

(last change: 30.08.2004)

(*)   University of Manchester
(**)   University Stuttgart, HLRS

Application Steering in a Collaborative Environment

John Brooke
University of Manchester
SVE, Manchester Computing
Manchester M13 9PL, UK
j.m.brooke@man.ac.uk

Thomas Eickermann
Research Centre Juelich
NIC-ZAM
D-52425 Juelich, Germany
th.eickermann@fz-juelich.de

Uwe Woessner
University of Stuttgart
HLRS
Allmandring 30a
D-70550 Stuttgart, Germany
woessner@hlrs.de

ABSTRACT

In this showcase we will present live running simulations which are integrated into the Access Grid in a variety of different ways. An example of this is the use of `vnc` to distribute a desktop on which the simulation is being displayed. Another example is the redirection of the visualization into `vic` to make 3D animations available over the Access Grid. Other examples that will be explored are the use of SGI's OpenGL VizServer to direct the output of a graphics supercomputer located on the Grid to the AG locations. We will also utilize the ability of the next generation AG software to directly link with visualization toolkits such as `vtk`, AVS/Express, or COVISE as an integrated part of the Virtual Venue as this functionality has developed by the time of the SC2003 demonstrations. We also demonstrate steering in a collaborative setting using a steering service which is fully compliant with OGSI and with the proposed OGSA architecture. This can be integrated with current Grid middleware (e.g. GT2 and UNICORE) using a specially developed Perl hosting environment, OGSI:Lite.

Keywords

application steering, Grid Computing, collaborative environment, Access Grid

1. INTRODUCTION

This showcase presentation will feature live steering of large scale computations viewed in multiple Access Grid nodes. Simulations in condensed matter physics, plasma physics, and fluid dynamics will be shown to audiences at the showcase floor and in other participating AG nodes. These simulations run on very large supercomputers and the collaborative research teams can be distributed across both intra and inter-continental networks. Since the resources in terms of computation and visualization are highly specialised and expensive to use, it is essential to utilise them as effectively as possible. Our usage scenarios are therefore

more akin to access to a large experimental apparatus such as a synchrotron or a telescope, than to the task-farming paradigm used in many Grid demonstrations. The aim of this paper and the associated presentation is to show how geographically distributed teams can view simultaneously a visualization of a running simulation and can steer the application. This utilizes the power of Access Grid [**?**] in being able to coordinate multiple channels of communication within a virtual space (the Virtual Venue of the meeting).

In order to computationally steer a simulation, one needs an interface to communicate with the simulation, which may be running on a remote machine. In addition to allowing parameters to be monitored and changed, this interface needs to offer the possibility of visualizing complex data sets, for instance 3D isosurfacing and volume rendering. To enable intuitive interaction with a simulation, it is essential that visualization can be performed sufficiently fast compared to changes taking place in the simulation. Visualization of large and complex data sets typically requires high-end graphics hardware, which, like high-end computing resources, is not always available locally. Therefore, visualization should be treated as a distributed resource as well, the need for which stems not only from computational steering but also from the high performance visualization community's needs. We present live running simulations which are integrated into the Access Grid in a variety of different ways. One example of this is the use of `vnc` to distribute a desktop on which the simulation is being displayed. Another example is the redirection of the visualization into `vic` to make 3D animations available over the Access Grid. Other examples that will be explored are the use of OpenGL VizServer to direct the output of a graphics supercomputer located on the Grid to the AG locations.

We also explore the ability of the next generation AG software (Version 2.0 and upwards) to directly link with visualization toolkits such as `vtk`, AVS/Express, or COVISE as an integrated part of the Virtual Venue. Currently this ability is still being developed. We therefore describe our applications running outside of an Access Grid framework as well as within this framework. In the former case, the view of the running application can be distributed via `vnc` from a screen attached to a machine integrated into the steering environment. This is not full integration into Access Grid but it does allow the advanced video-conferencing capabilities to be linked with the steering of the application and allows for wider participation in the experiment. Specifically, some of the AG sites may be unable to participate in the steering but can view the output from the steering experiment as it
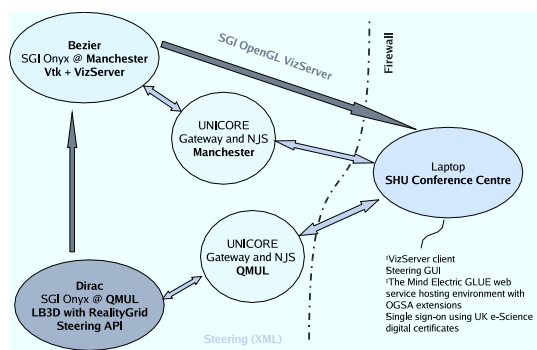
Figure 1: The prototype RealityGrid steering application operating as a Web service. Computation and visualisation are on different machines and the steering and visualisation can be viewed and controlled from a users laptop.

evolves.

2. REALITYGRID

2.1 Aims of RealityGrid

One demonstration will be from the RealityGrid project [?] which aims to predict the realistic behavior of matter based on the properties of the microscopic components using diverse simulation methods (Lattice Boltzmann, Molecular Dynamics and Monte Carlo (e.g. [?, ?]) spanning many time and length scales and the discovery of new materials through integrated experiments. A central theme of RealityGrid is the facilitation of distributed and collaborative exploration of parameter space through computational steering and online, high-end visualization. A RealityGrid scenario involves a large-scale simulation running on a massively parallel system at on site coupled to a high-end visualization system at another site with the steering and display interfaces running at one or more remote sites. The simulation component periodically (or as demanded by the steerer component) emits "samples" for consumption by the visualization component, while grid middleware is responsible for the transfer of data between components. The primary AG node will be the University of Manchester but the computation and visualization will be distributed over different supercomputers in both the US and Europe.

2.2 Computational Steering in an OGSA framework

An important intention of RealityGrid is to develop computational steering within the framework of the Open Grid Services Architecture [?].The steering client, i.e. the part that can be integrated into the collaborative environment, contacts a steering service which will actually orchestrate the details of the steering and associated visualization (Figure 1). A prototype of such a steering service was demonstrated at the UK e-Science All-Hands meeting in September 2002 [?]. This utilised a prototype Grid service implementation posted to the OGSA mailing list in March 2002 by

Dave Snelling of Fujitsu. The Web service was contacted from a laptop on the conference floor. The orchestration of the compute and visualization servers and the file transfer was handled by UNICORE (UNiform Interface to COmputing REsources) (for details see chapter 3.1, [?, ?]) using the Unicore Protocol Layer. This allowed the application to simulate the behaviour of a possible OGSA service before the OGSI working group had formulated its standards recommendations.

The computation was a Lattice Boltzmann 3D code simulating a mixture of two fluids. The parameter used for the steering was the miscibility of the fluids. The simulation was on a 3D grid with periodic boundary conditions. As the miscibility parameter was altered, the structures formed by the fluids changed and the visualization was necessary so that these changes could be observed. It is this visual element to the steering that necessitates the incorporation of a visualization supercomputer into the workflow. In the original demonstration the computation was carried out on an SGI Onyx (Dirac) at UCL in London. Samples were emitted as the computation progressed and sent over the UK SuperJanet network to another SGI Onyx in Manchester (Bezier). The isosurfaces were rendered and the output of the graphics pipes returned to the users laptop at the Conference floor in Sheffield by SGI OpenGL VizServer. It is purely coincidental that in this actual example the compute and visualization servers have the same architecture. No such restriction is inherent in the middleware driving the application.

The application could traverse firewalls since the UNICORE architecture places security Gateways at the firewall boundary. The workflows being instantiated are known in UNICORE as Abstract Job Objects (AJOs) and are sent via `ssl` as serialised Java objects. They are received by a Network Job Supervisor (which may or may not be on the same machine as hosts the gateway) and the AJOs are translated into Perl scripts for a target machine. This proeess is known as incarnation in the UNICORE model; it allows the details of the scripts used to run the workflow to be hidden from the application. This is a very important part of the process of abstraction necessary for the creation of Grid services.

2.3 An OGSA Steering Service

Since then the steering application has also been implemented using Globus Toolkit version 2.4 as part of the UK Level 2 e-Science Grid [?]. The experience gained from these implementations is now being incorporated into the development of an OGSA Computational Steering service. The proposed architecture is shown in Figure 2.3. The laptop used in Figure 1 is now replaced by the abstraction of a Steering client. This contacts a registry which have details of the steering services that have published to the registry. For illustration we show one service that steers the application and another that steers the visualization. In more complex workflows there could be more services, possibly linked to live experiments or database searches. The steering services allow all of these components of the workflow to be steered. This shows the potential of the OGSA approach. The client chooses the services it will require and binds them to the client. From the point of view of enabling applications, the RealityGrid project has defined APIs for the steering calls which can be used to link from the application to the services. Some workflow description is necessary
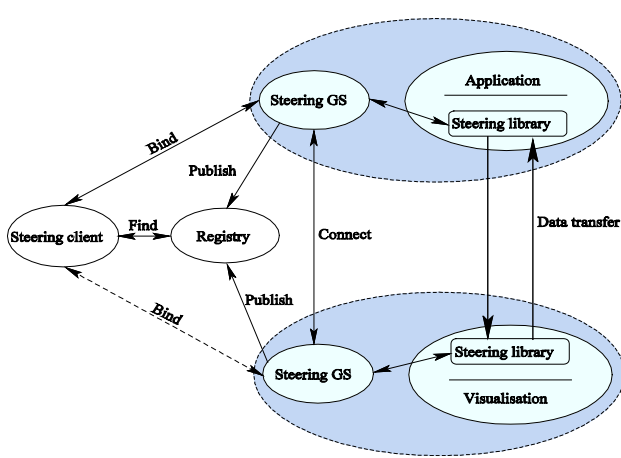
Figure 2: OGSA steering service architecture, implemented in the RealityGrid steering service

to connect the services.

Currently the very first implementations of the proposed OGSI standard are only becoming available (GT3 and .NET implementations announced in July 2003). These early implementations have very basic functionality, insufficient for our steering application. Moreover, the RealityGrid project wishes to continue to utilise the UK Level 2 e-Science Grid which will be based on GT2 until 2004. RealityGrid has therefore developed a lightweight OGSA hosting environment called OGSI-Lite [?]. This uses Perl to create the hosting enviroment and can thus run on almost any platform. In fact part of the testing platform for OGSI-Lite includes a Sony Playstation 2. This echoes the structure of the original UNICORE implementation of the steering service since the UNICORE TSI (Target System Interface) uses Perl to incarnate the Abstract Job Object. Thus UNICORE fits well in the role of a precursor of OGSA and the ideas of the original architecture continue to be applicable and updatable in a Grid Services framework.

2.4 Collaborative Aspects

The RealityGrid demonstration will show the Lattice-Boltzmann computation being run on a remote supercomputer. The visualization will also be rendered on another separate supercomputer as shown in Figure 1, however we utilise the AccessGrid to replace the single client with participating clients at multiple sites.All participating sites who have native multicast enabled will be able to view the visualization, this can be described as passive collaboration. Participating sites able to run OpenGL VizServer will be able to share control of the visualization using VizServer's collaborative abilities. Sharing the steering client requires the use of `vnc`. This is the active mode of participating. Here we go beyond the advanced video-conferencing capabilities of the Access Grid, moving towards a genuine collaborative problem-solving environment.

Collaborative visualization is also achieved by means of the `vtkNetwork` extension to `vtk` provided by the Futures Lab, Argonne National Laboratory, University of Chicago. This package provides a specialised `vtk` rendering class which streams updates to its framebuffer to a multicast address. Remote users can then view the broadcast visualization

through a standard `vic` session.

The vtkNetwork classes also allow for collaboration by end users, by sending any remote events back to the visualization application using a patched version of `vic`. We do not take this approach however, and instead use the collaborative features offered by VizServer, which allows multiple users to share the same login session on a remote machine. This removes the need for any end user's wanting to participate in a collaborative visualization having to patch their version of `vic`.

We draw attention to the power of tools such as VizServer. The datasets which are being rendered as isosurfaces are too large to be visualized on a laptop client. VizServer allows the output of the graphics pipes from an Onyx visual supercomputer to be accessed remotely. In addition this greatly reduces network traffic since only compressed bitmaps need to be sent to the participating sites. We believe that this is a realisation of one of the common claims for the utility of a Computational Grid, namely that it provides seamless access to resources beyond the capabilities of a desktop or laptop machine. Moreover, as described in Sections 2.2 and 2.3 above, we have integrated this seamlessly into the collaborative session, the participants need never know details of which supercomputers are running the application. Moreover RealityGrid is developing the ability to migrate both computation and visualization within a session without any disturbance or intervention on the part of the participating clients. Thus the Grid is used as a form of computational power, i.e. the resource per unit time needed to run the simulation and visualization at a pace that keeps the interactivity at the client level. This is the novelty of this demonstration indicating how far Grid services have moved beyond the original batch submission, or task farming model.

3. STEERING UNICORE APPLICATIONS WITH VISIT

Research Center Juelich is the primary AG node for the second demonstration which will present a steering extension to the UNICORE Grid system based on the steering toolkit VISIT. As an application, a new plasma simulation code, PEPC (Parallel Electrostatic Plasma Coulomb-solver) is shown. All participating AG sites will be able to share the experience via `vic` and `vnc`. In addition to that, sites that have AVS/Express installed can actively participate in the collaborative steering of the application. In the following sections, the UNICORE system, VISIT, PEPC, and the planned demonstration will briefly be described.

3.1 The UNICORE Grid System

The UNICORE software provides a Grid infrastructure together with a computing portal for engineers and scientists to access supercomputer centers from anywhere on the Internet. In contrast to e.g. the Globus [?] toolkit, the most widely deployed Grid software today, UNICORE, which was developed in several German and European projects [?, ?, ?], follows a vertically integrated approach. It offers seamless and secure access to distributed computing resources, with a focus on workflow management of batch jobs. Besides its intuitive user interface, an advantage of UNICORE is that it does not require any modifications of the applications that run under its control. UNICORE is used as a "production system" in several German HPC centers and

has recently been selected as the Grid middleware for the new Japanese National Research Grid Initiative (NAREGI) [?].

The UNICORE Grid system consists of three distinct software tiers:

- UNICORE client interacting with the user and providing functions to construct, submit and control the execution of computational jobs,

- UNICORE servers that are divided into gateways acting as point-of-entry into the protected domains of the HPC centres and Network Job Supervisors (NJSs) that adapt the abstract UNICORE job for the specific HPC system,

- UNICORE target systems that schedule and run the jobs on the HPC platforms. On these systems a Target System Interface (TSI), which is available as a Java application or a set of Perl scripts, performs the communication with the NJS.

Here we present the design and a prototype implementation of an extension to UNICORE that supports computational steering. It is based on the steering toolkit VISIT [?] and is designed to preserve the following strengths of UNICORE:

- single sign-on with strong authentication and encryption,

- firewall-friendliness; handling of all communication over a single fixed TCP server-port,

- immediate portability; any application that uses VISIT will be able to use the VISIT-UNICORE extension without modifications.

The only component of the UNICORE system that needs to be modified for this extension is the TSI, thus maintaining backward compatibility with the standard UNICORE system.

3.2 The Visualization Interface Toolkit (VISIT)

The VISualization Interface Toolkit (VISIT) is a lightweight library for online visualization and computational steering. Its first version has been developed in the project 'Gigabit Testbed West' [?], a testbed for the German Gigabit Science Network, the G-WiN. Since then, VISIT has evolved into a stable software used in several application projects, mainly on the Supercomputers of the John von Neumann Institute for Computing (NIC) at Juelich. A main design goal of VISIT was to minimize the load on the steered simulation and to prevent failures or slow operation of the visualization from disturbing the simulation progress. This means that all operations (like opening a connection, sending data to be visualized or receiving new parameters) have to be initiated by the simulation and are guaranteed to complete (or fail) after a user-specified timeout. This led to the design decision to implement VISIT as a simple client-server application where the visualization acts as a server that dispatches the simulations requests - unlike many other steering toolkits that work the opposite way [?, ?, ?, ?, ?]. To keep VISIT portable to 'classic supercomputers' which often lack good implementations of common UNIX protocols or tools, the simulation side of VISIT in particular does not rely on any external software or special environment and has a lean and easy-to-use interface.

VISIT uses an MPI-like data transport mechanism based on messages that are distinguished via tags to transfer simple data types like strings, integers, floats, user defined structures, and arrays of these. The client either sends data along with a header describing its content or requests data from the server by sending a header that describes what is requested. Any data conversions (byte order, precision, integer-float) are performed transparently by the server, again so that the simulation is disturbed as little as possible. The client (simulation) part of VISIT has C, Fortran, and Perl language bindings. On the server (visualization) side VISIT supports C, Perl and AVS/Express, a commercial visualization software. Bindings for Java, IDL (Interactive Data Language) and tt vtk are currently being developed. VISIT is freely available for download [?] in source-form for common UNIX platforms and Microsoft Windows.

A major drawback of VISIT is that it does not provide any encryption or other means of security except for a connection password that is transferred in clear-text. Due to its dynamic TCP-port selection scheme it also does not work well with firewalls. These problems are resolved by the integration of VISIT with UNICORE which is described in the next section, since UNICORE offers exactly the features needed.

3.3 The Steering Extension to UNICORE

The main technical challenge when trying to steer VISIT applications running under control of UNICORE is to map the different communication models of these two systems correctly. In UNICORE, the user uses a client to submit jobs, query their status and eventually fetch the results. All these operations are separate transactions that do not require a stateful connection between the UNICORE client and the target system to be maintained. While this contributes to the robustness of the UNICORE system (a client can appear or vanish at any time) it does not match the connection-oriented architecture of VISIT well, where the steered application is the client and the visualization the server.

To overcome that we have designed and implemented a connection-oriented protocol on top of the UNICORE protocol. The simulation-end of that connection is formed by VISIT proxy-servers which are separate processes running on each target system. The other end of the connection is located at the UNICORE client, implemented as a client-plugin and acting as a VISIT proxy-client. By polling the target system for new data, that plugin is able to emulate the server capabilities that are required for the VISIT connection. Using the proxies at the TSI and UNICORE client, VISIT applications can communicate without having to be modified.

With the VISIT extension to UNICORE described so far, a user can attach one online-visualization or steering application to a UNICORE job at a time. However, a further modification to allow multiple users to collaboratively view and steer a simulation is straight-forward. To achieve this, the simulation data has to be sent to all visualization applications and these applications have to exchange information among each other to ensure that everyone has the same view of the data (e.g. position and orientation of view point

or parameters like thresholds that influence the visualization). Since the simulation acts as a client in VISIT, the former task can easily be implemented by a 'multiplexer' that simply sends all VISIT send-requests to all participating visualizations, ensuring that everyone views the same data. Receive-requests are only sent to a 'master' visualization, so that only that master is able to actively steer the application. The master-role can be moved between the simulations allowing for a coordinated cooperative steering. This functionality has been implemented in an application (the vbroker) that is part of the standard VISIT distribution. For the VISIT-UNICORE extension this functionality has been moved into the VISIT proxy-server running on the UNICORE target system. This has the advantage that all users participating in the collaboration have to authenticate to the UNICORE system.

Like video and audio, the exchange of control information between the visualizations is sensitive to latency if a 'sense of presence' is to be created among the users. Therefore we do currently not use UNICORE communication mechanisms for that purpose. Instead, we have implemented an external server that collects and redistributes the control data. This server allows to assign different roles to the participants: one role allows to change visualization parameters like the view angle and a second role is just for passive viewers.

3.4 The Access Grid Demonstration

The application that is used for the demonstration is PEPC (Parallel Electrostatic Plasma Coulomb-solver), a new plasma simulation code [**?**]. PEPC has been instrumented to be steered with VISIT independently of the VISIT-UNICORE extension and does not necessarily require UNICORE for its operation. The code uses a hierarchical tree algorithm to perform potential and force summation for charged particles in a time $O(N \log N)$, allowing mesh-free particle simulation on length- and time-scales normally possible only with particle-in-cell or hydrodynamic techniques. The VISIT interface was integrated at a very early stage in the development of PEPC to assist in verifying the correctness of the tree structure and domain decomposition for geometrically complex systems: for example, a particle beam striking a spherical plasma target. This is done by regularly shipping both particle data-space comprising coordinates, velocities, charge, processor number and tracking-label plus information on the tree structure, at present consisting of a set of node coordinates representing each processor domain. Particles are displayed as points, diamond glyphs and vectors, including time-histories over several time-steps; tree domains as transparent or solid boxes, providing immediate insight into both the physical and algorithmic workings of the parallel tree code. A future extension will also provide selected diagnostic quantities mapped onto a user-defined mesh, such as charge density, current, electric fields and laser intensity.

A further implemented feature is a capability, in which the particle beam or laser parameters (charge/intensity, direction) can be altered by the user interactively while the application is running. This is particularly useful in verifying start-up parameters such as the initial alignment of laser and target, or for performing quick trial runs with reduced number of particles as a prelude to full-blown production. Another example where this capability can save a lot of trial-and-error run-time is in determining equilibrium plasma states for arbitrary geometry (slab, spherical,
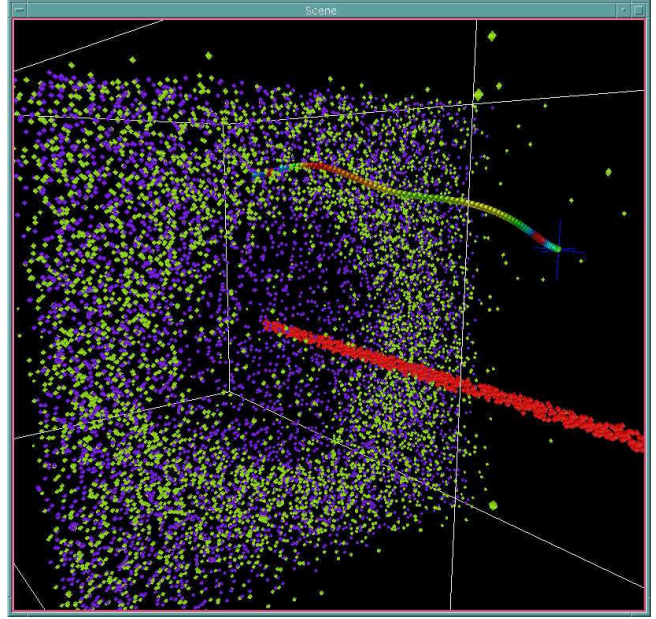

Figure 3: VISIT-based online-visualisation of a PEPC simulation; a plasma hit by a particle beam.

cylindrical etc.). Thus, the user can 'assist' an initially random plasma system towards a cold, ordered state suitable for use as quiescent initial conditions in a laser interaction simulation.

A demonstration of the prototype implementation of the VISIT extension to UNICORE will be the main part of the presentation. A parallel simulation of a laser-plasma interaction with PEPC (see Figure 3.4) will be started under control of UNICORE. Intermediate results will be visualized with the AVS/Express based steering application, allowing parameters of the running simulation to be modified on the fly. All Access Grid Constellation, Satellite and Observer Sites will be able to watch the demo without having to install any additional software. The UNICORE client and the AVS/Express control panel will be made available via `vnc` while the AVS/Express 3D visualization will be fed into a standard `vic` video stream. If feasible, the collaborative features of the system will be demonstrated, too. However, in order to take part in that demo, an Access Grid node will have to install AVS/Express. Those nodes could take part as passive viewers; the AG node in Phoenix would be granted full access.

4. INTEGRATION OF A COLLABORATIVE ENVIRONMENT WITH ACCESS GRID

The third demonstration, contributed by HLRS, will be based on the integration of the collaborative visualization and simulation environment COVISE [**?**] with the Access Grid. We first discuss the design decisions of such an environment before explaining its usage potential when integrated with the Access Grid.

4.1 Requirements on a Collaborative Environment for Application Steering

Collaborative working applied to the steering of ongoing

simulations is a highly interactive process that requires fast turn around times of multiple feedback loops. It is assumed, that not only the simulation is performed on a remote high performance computer but also post processing steps up to the rendering of images might be executed in a distributed environment. The design of the distributed software architecture has a strong influence on its characteristic behaviour regarding time delays on interactions, responsiveness in a collaboration process but also scalability with increasing volumes of data.

4.2 Reaction Times of the Rendering Feedback Loop

The highest demand on the reaction time is given when visualizing simulation results in a virtual reality environment such as a CAVE. When a user moves, the whole scene content has to be redrawn from the perspective of the new viewer position with at least 10 to 15 updates per second. In case of a remote rendering the new viewer position first has to be transmitted to the rendering side where the new image is generated, compressed, transmitted back to the viewing station, decompressed and finally displayed. Just taking the communication delays as well as the compression and decompression times into account, without considering the rendering times, these already exceed the required turn around time. Therefore typical distributed virtual environments work with local scene graphs using local graphics hardware for rendering. For collaboration in a distributed virtual environment the positions of participants are sent out. In a local scene display the other participants are represented by avatars. Thus it is barely noticeable if a delay in updating an avatar position appears.

When using a desktop workstation for visualizing the content the requirement on maximum delay until the scene is rerendered from a new perspective is less demanding than in a virtual environment. At least 3 to 5 frames per second should be reached with one frame delay to react on scene interactions. In a collaborative session it is expected that all participants share the same viewer position providing the same content as a basis for discussion. A variation of one frame does not influence a discussion process, while multiple frames difference in the discussed visual content might lead to misunderstanding and thus result in unusable working conditions. Taking large volumes of time dependent simulation data into account as well as larger variations in networking bandwidth and delays for different participants in a collaborative session such differences in currently visualized content become very likely.

Therefore a group collaboration environment for high end simulation steering needs to handle such synchronization issues.

4.3 Reaction Times of the Post Processing Feedback Loop

The collaborative analysis of a simulation is an explorative process which requires modifying parameters of a visualization tool such as a cutting plane position or apply different tools in the evolving exploration. Collaborating partners always need to have the same state of information about the overall system and need to be able to change roles, i.e. actively steering the exploration process or passively watching but participating in the discussion. The delays until a parameter change in a visualization tool leads to an updated scene content can vary strongly and be in the range of parts of a second to multiple seconds. The more stringent requirement here is, that the update takes place at the same time at the different participating sites of a discussion. With a local feedback loop involving the generation of a new cutting plane and rendering it, depending on the interaction in a virtual environment, it is possible to have 15 or more frames per second with modified content. In a collaborative environment such scene update rates are only possible if the generation of the new content is done locally and only synchronisation information such as the parameter set for the cutting plane determination is exchanged.

4.4 Reaction Times of the Simulation Feedback Loop

The still acceptable delay on the modification of simulation parameters is defined by the time a human being is able to stay mentally in the model world of the simulation without noticing any reaction or activity of the system. Experiments showed that people can tolerate delays of up to a minute while waiting for new simulation results. This tolerance can even be increased if intermediate results like from an iterative solver are displayed in-between. For outstanding actions that don't show an effect over tenth of seconds or more the scientist needs a visual reminder that there are still ongoing activities. A common approach is the usage of the hourglass icon for the cursor or an indicator for the remaining time to wait. Also here it is required for a collaborative session that the modified visual content appears synchronously to prevent discussion on inconsistent content.

4.5 COVISE Characteristics

COVISE has been developed since 1993 in a series of European projects initially in collaboration with aeronautics industry and research organizations. Together with the automotive, aerospace, and mechanical engineering industry it has been applied as an integration platform for the simulation process chain covering grid generation, simulation and post processing. The common characteristic of these projects was, that distributed engineering groups work collaboratively to avoid their relocation.

Thus the design criteria for COVISE were to support collaborative working of specialists analysing simulations performed on remote supercomputers. A distributed software architecture has been developed that allows to execute and control process chains involving multiple computers which extends beyond a client server approach. By integrating simulation and visualization into one homogeneous environment and controlling all distributed processes from one user interface, simulation steering is inherently available from the outset. COVISE has especially been optimized for efficient network transfer [?] and high performance computing environments [?]. The user interface is based on the visual programming paradigm as used also in other visualization packages such as e.g. AVS. Distributed applications can be built by combining modules (modeled as processes) from different application categories on different hosts to form module networks. At the end of such networks the rendering step performs the final visualization. This application building step is done in the Map-editor module, the central user interface of COVISE. Session management for adding new hosts and synchronizing the tasks in the module network is done in a central controller which has the only

knowledge about the whole application topology. Request brokers on each participating host take care of data management, efficient data transfer and conversion between different platforms. COVISE in contrast to other visualization systems uses the notion of data objects instead of relying on a pure data flow paradigm. The underlying data management takes care of assigning system-wide unique names to data generated during a session in the shared data spaces: the shared data space (SDS) is used on a single host for the exchange of data objects between the locally running modules to minimize copying overhead. On most platforms this is realized as shared memory communication. Between heterogeneous hardware platform data type conversion is done by the request brokers which is thus invisible for the application modules. Scientific data is handled as data objects which have attributes such as names and lifetime. They represent grids on which dependent data is defined. Dependent data such as temperature or velocity themselves are again data objects.

In a collaborative session all partners see the same screen representations at the same time on their local workstation. The results of the visualization as well as user interactions are displayed in a synchronized way at each site. Visual interactions with steered applications are handled in the same manner. Standard audio/video conferencing tools such as `vic` and `rat` and shared whiteboard tools have always been used for a discussion support.

4.6 COVISE and Access Grid

As we already had developed a collaborative steering environment that also extends towards virtual reality usage, there was no need for a Grid based new development. Video and Audio conferencing with simple web cams and PC based microphone and loudspeaker proved to be a major week point in many of the collaborative working oriented projects. Therefore the integration of COVISE with the Access Grid, where the audio/video conferencing issues are mainly resolved, provides a major benefit for COVISE.

During 2002 a special venue server compatible to Access Grid 1.2 has been implemented that allows to start application sessions such as COVISE consistently within the Access Grid group collaboration sessions. This venue server stores additional information on a per room basis which allows the start-up of shared applications. Other than videoconferencing environments, virtual environment systems are often behind firewalls which do not support multicast and sometimes even do NAT. Thus, we added support for unicast/multicast bridges and point to point sessions.

As COVISE also supports collaborative sessions in virtual reality it becomes possible to combine distributed virtual environments with Access Grid discussions. For this purpose a stereo power wall was split into a stereo projection part with magnetic tracking support for steering and an Access Grid part for seeing the participating sites. In Figure 4.6 the passive stereo glasses have been taken off to take the photos.

It is obvious, that COVISE needs to be installed on all participating sites of a session, which can be regarded as a draw-back compared to a `vnc` based sharing approach, where the application is not aware, that a collaborative session is going on. On the other hand this approach allows a much better scaling in the handling of large volumes of scene content such as coming from complex simulations. Additionally

Figure 4: Access Grid session with collaborative virtual reality based discussion

the collaboration speed does not degrade with the volume of displayed geometric data.

Currently the effort to port the functionality available in our existing venue server to the actual version of the Access Grid is determined. It is intended to have the collaborative steering available with Access Grid 2.0 for the demonstrations at Supercomputing 2003.

4.7 Application Demonstration

The demonstrations will be based on a Car Show and Brand Representation Building. Simulations allow determining and optimizing the climatization layout of such a building. In collaborative visualizations architects, managers and engineers performing the simulations are able to discuss the building layout and its implications on the climatization. Furthermore the behaviour of visitors of such buildings will be simulated and analyzed. In collaborative sessions it will be possible to discuss alternative building and exhibition layouts to steer the visitors and potential customers into certain regions of the building. The demonstrations represent a collaboration between HLRS and DaimlerChrysler who contributed the architectural design process for their Auto Houses as well as with Sandia National Labs related to the behavior of people in complex building structures. Architectural discussions in distributed virtual environments such as coupled CAVE-like environments are performed already now. The integration into the AG should enhance the group collaboration capability and make such technologies accessible to a larger community.

5. CONCLUSION

The integration of the computational and visualization power available over the Grid will be the main contribution. This extends the collaboration side of the Access Grid to incorporate the modeling of reality via simulation. When several researchers are located in the same visualization theatre or clustered around a workstation, collaboration is trivial - we do not even need to think about the protocols involved as they have become second nature over a lifetime of interac-

tion with other human beings. Without co-location, the loss of human-human interaction seriously impairs the collaborative experience. With substantial effort, it is possible to introduce software that partially replaces these human-human interactions with machine-mediated mechanisms. However, Access Grid technologies link separate locations into a virtual environment, effectively re-instating the audio and visual inputs on which human beings are so dependent. We can therefore go a long way towards creating an effective environment for distributed, collaborative computational steering and visualization by integrating our user interfaces with Access Grid. The integration of a collaborative visualization and simulation environment, that scales from desktop user interfaces to immersive projection environments, with the AG, will enable group collaboration usage modes that also range from monoscopic large projection walls to stereoscopic immersive hardware set-ups. Also heterogeneous setups are supported such as the coupling of typical AG rooms with AG enabled CAVEs. Working in such environments should improve the understanding of its influence on the group collaboration process. Human factors issues in discussion processes have to be analyzed for such setups. SC2003 demonstrations will showcase the coupling of remote AG enabled CAVE like environments with show floor environments.

We consider that this collaborative aspect of the Grid has been somewhat eclipsed by the data grid and task farming applications which currently seem to dominate Grid computing. Examples are multiple task farming of gene-sequencing algorithms, parameter space searches, analysis of huge volumes of data from large-scale particle physics experiments. These are very valuable uses of Grid computing, but they are not the only possible ways to utilise Grid resources. Here we show an alternative use, namely the incorporation of specialist massively parallel supercomputers and specialised visual supercomputers into a distributed collaborative working session.

We have shown also that this can be done in the context of the proposed Open Grid Services Architecture Framework. The RealityGrid demonstration, in particular, is run as an OGSA-compliant Grid service. We are even able to run this using Globus Toolkit 2 and UNICORE which are pre-OGSA systems, by the creation of a very lightweight hosting environment, OGSA:Lite. This will enable applications currently running on the many Grids that currently utilise GT2 and UNICORE to explore the potential of the OGSA framework which the full OGSA systems such as GT3 are being developed.

Finally, we have shown that the feeling of live human presence captured by Access Grid can co-exist with the use of the Computational Grid to access large scale resources. This brings supercomputing to the collaborative distributed meeting environment and will provide a new and exciting way for geographically distributed research groups to improve their scientific productivity. We can speculate that in the not too distant future, a major scientific breakthrough will be achieved in a live collaborative session. Then, one could say that the use of the word Grid in Access Grid is fully realised.

6. ACKNOWLEDGMENTS