“CoDeveloper: A secure, Web-Invocable Collaborative Software Development Tool”

Grant No DE-FG03-00ER83311

Final Report

Svetlana Shasharina (PI)
Tech-X Corporation
sveta@txcorp.com

1. Introduction

Modern scientific simulations generate large datasets at remote sites with appropriate resources (supercomputers and clusters). Bringing these large datasets to the computers of all members of a distributed team of collaborators is often impractical or even impossible: there might not be enough bandwidth, storage capacity or appropriate data analysis and visualization tools locally available. To address the need to access remote data, avoid heavy Internet traffic and unnecessary data replication, Tech-X Corporation developed a tool, which allows running remote data visualization collaboratively and sharing the visualization objects as they get generated. The size of these objects is typically much smaller than the size of the original data.

For marketing reasons, we renamed the product CoReViz. The detailed information on this product can be found at <http://www.txcorp.com/products/CoReViz/>. We installed and tested this tool at multiple machines at Tech-X and on seaborg at NERSC. In what follows, we give a detailed description of this tool.

2. Motivation for Design

The scientific community is facing increasingly difficult problems data analysis due to the size of the produced datasets and the need to have those datasets analyzed and visualized at multiple research sites. As an example, we consider a now common plasma simulation on a three-dimensional grid with 2,000x200x200 points. In a fundamental electromagnetic simulation, one must save at least 10 values (three components of electric and magnetic fields along with density and flow velocity for a single fluid) per grid cell. This implies that the data in a single time slice is 6.4 GB. In particle simulations, the problem is much worse – one can easily have 5 particles per cell, with each particle described by six or more variables. For such simulations, the size of the data in a single time slice is now 26 GB, and one can generate multiple time slices (easily 30 or more) per run. Hence, the data produced by a single 3D simulation easily approaches a terabyte.

Transferring such amounts of data can be problematic: it requires appropriate transfer mechanisms, like GridFTP [1] or Logistic Networks [2]. But even with the appropriate transfer mechanisms, streaming raw data [3], as it is being produced by simulations, is not always an adequate solution, because the client might use high-latency and low-bandwidth connections or not have enough storage. Moreover, it often makes sense to analyze and filter data before bringing it over, since not all of it is relevant for generating new knowledge and might be discarded.

In addition, one cannot count on data being needed at one place only. Increasingly, simulation efforts are multi-institutional, with participants who are widely geographically distributed. One example is the NIMROD collaboration [4], which involves participants from SAIC and General Atomics in San Diego, the University of Colorado, the University of Wisconsin, and other institutions. Another example is the M3D effort [5], which involves participants at Princeton Plasma Physics Laboratory and New York University. The VORPAL [6] team

includes researchers from the University of Colorado, Tech-X Corporation and Lawrence Berkeley National Laboratory. In each of these collaborations, data is generated at a remote supercomputer, while the data analysis and visualization scripts might come from any of the participating institutions.

The design of CoReViz is based on our experience of working with such distributed physics collaborations. For example, Tech-X plasma physicists routinely run their NIMROD and VORPAL simulations at NERSC [7]. The generated HDF5 [8] data gets stored at the scratch area of a NERSC supercomputer. Then the scientists use multiple tools such as Interactive Data Language (IDL) [9], SCIRun [9] or OpenDX [11] to extract particular subsets of data and create 2D or 3D objects, which they then manually bring for local viewing and possible subsequent movie generation.

CoReViz partially automates such workflows and provides an integrated environment for data visualization, allowing for multiple visualization tools at the back-end. It allows users to leave the raw data at the location where it was originally generated, automatically collocate data analysis and visualization routines coming from multiple locations with the data, run them on the data, and automatically distribute the visualization results to multiple collaborators. The visualization objects and the data analysis and visualization scripts are much smaller than the sizes of raw data, thus the network is not overburdened.

There are several tools which provide remote visualization. Some of them are client-server based (Ensign Gold [12], Visual Area Network [13] and OpenDX [11]) while some use the X Window System over WAN (gnuplot [14], ImageMagick [15], IDL [9] and AVS/Express [16]). Since X Windows is very verbose, doing remote visualization using X Windows over WAN is prohibitively slow. Also because OpenDX does not support collaboration, only Ensign Gold and Visual Area Network (which runs only on SGI) are of competitive interest to us. Both Visual Area Network and Ensign Gold are comprehensive and expensive commercial products with the abundance of features for commercial visualization and are somewhat overkill for the most scientific applications.

CoReViz is more limited as far as visualization capabilities are concerned, but allows users to run multiple third party scripts and partially automates workflows common for large scientific simulations. Although CoReViz uses client-server architecture, it does not need installation on the client side (Java jars get downloaded through a browser). Finally, CoReViz is “economically” priced as it targets the scientific community, not large commercial companies.

The remainder of the paper is organized as follows. Section 3 describes the design and implementation of CoReViz along with the technical details. Section 4 provides information on the use of CoReViz. Section 5 describes movie generation feature in CoReViz. We conclude with the summary and some ideas for the future.

3. CoReViz design

3.1. Overview

CoReViz is an application developed at Tech-X Corporation, which allows users to run remote data analysis and visualization in a collaborative manner, to share the visualization results and render the visualization objects locally (Figure 1).

Figure 1: CoReViz allows remote data analysis and visualization in a collaborative manner.

It is a client-server application with a Java client, C++/Python server, and CORBA middleware, as shown in Figure 2. CORBA is used to send messages to the server but not to transfer inputs and outputs (just to trigger the transfers). On the server side CoReViz uses TAO [17], and on the client side, CoReViz uses JacORB [18]. Both TAO and JacORB are free ORBs, which facilitates their use within the scientific community. All the CORBA calls are tunneled through *SSH* (secure shell), making calls secure. The input files (visualization and data analysis scripts) and the output objects are sent via *SCP* (secure copy). The server talks to visualization applications using UNIX pipes.

Figure 2: The main elements of CoReViz: the client communicates to the server using CORBA. The server talks to applications through UNIX pipes. The transfer of files is done via SCP.

2.2 Implementation

The server is written in C++ and Python. C++ is used as a CORBA server, while Python is used as glue between the C++ server and visualization applications (for example, IDL). The server also utilizes a standalone Java tool, which is used for quick-time movie generation. The server's responsibilities include listening to clients, serving as a connection point between them, running applications on the data located on the server and notifying clients about newly generated visualization objects.

The assumption in our design is that the users of CoReViz have a login to the server machine and authenticate themselves using their passwords. When a user starts the CoReViz server, she actually starts a new CORBA server process on the remote machine. When she starts a new CoReViz session for running visualization applications, the

server forks the application that is to be served in a new process (per the instructions in the configuration file) and connects the standard input and standard output of the new process to inter-process communication (pipes) that the server can read from and write to.

After the session has started, the user can open a local or remote visualization script and run it on the server. The results of such visualizations can be the following:

- Generation of files with extensions png, jpg, and gif for 2D visualization objects.
- Generation of VRML files for 3D visualization objects.
- Changes in remote X windows due to new images generated by the visualization applications acting as X clients.

To deal with the first two types, the server performs regular scanning (1 second by default) of the directory specified by the first connected client to see if new files of interest have appeared or existing files have been changed and sends notification to all connected clients, who then can use SCP to bring the new files to their local machines.

If the visualization application opens an X window to display results, we use the following strategy. Every new CoReViz session starts a lightweight X server, called Xsave, which is a modified Xvfb server [19]. Xvfb takes advantage of the Damage extension of X11R6.8.2, which allows the server to be notified whenever something “significant” is drawn to the window, and of the Composite X extension redirecting the images to the off-screen buffer and bringing images of overlapped windows. Our modifications to Xvfb allowed the X clients (the applications) to avoid connecting to this server to get the images and to dump images to JPEG files into the monitored directory.

The CoReViz server detects these new files (since files with extension .jpg are in the list of the files which are watched), notifies the CoReViz clients, and they then use SCP to bring the new files, which happens without user intervention. The described method is faster than using X over WAN, because it avoids verbose WAN communication between the X client and server (X client and X server are collocated on the CoReViz server) and sends compressed files (.jpg) upon receiving notifications.

Although currently we support only shell and IDL scripts as visualization applications on the server, we have also experimented with Python, OpenDX, and gnuplot sessions, which could be added to the production version of CoReViz, if requested.

The CoReViz client is implemented in Java and allows users to choose between the types of sessions (single-user versus multi-user, File-Handling versus IDL), view, edit and share local and remote text and image files, and collaborate via chat and image markups. The VRML rendering uses Jun4Java [20] - open source software allowing for displaying and manipulating VRML files. Like 2D files, VRML files can be shared. In addition, the client provides means to trigger creation of movies from specified 2D files on the server (see Section 4 for details). Finally, the client uses SCP to get remote visualization objects upon server notification.

4. Using CoReViz

4.1. Connection manager

CoReViz usage starts by going to a CoReViz client website, currently located at <http://coreviz.txcorp.com/>. The CoReViz client is available at this site as a Java Web-Start application, which allows downloading the needed jar files and opens the CoReViz connection manager shown in Figure 3. Using the connection manager, one can choose a remote URL on which to start a server, or connect to it if it is already started, and the port number where the server runs currently. CoReViz is installed on several machines, including Seaborg, an IBM supercomputer at NERSC. One can also choose between the multi-user and the single-user modes (in the single-user mode, all collaborative features of CoReViz are disabled).

Starting or connecting to a server requires a user to have a login to the server machine, which is why the connection manager has entries for the username and password. If the user chooses the single-user mode, the connection manager allows also choosing the type of the session. In the multi-user mode, the choice of the session is postponed till the main client window opens. This is done because the user might want to join existing sessions rather than starting her own. In the next sections, we give more detailed descriptions of the single-user and multi-user modes and types of sessions they support.

Figure 3: The connection manager allows choosing the URL and port number for the server, starting or connecting to the server, and choosing the mode and the type of the session.

4.2. Single-user mode

The single-user mode currently allows two types of sessions: the File-Handling and IDL sessions. After the user chooses the type of session, she is presented with the session window (see Figure 4). This window allows the user to enter or select the path on the remote server using a remote file browser to a directory, which the server will monitor for new files. The further choices (handling 2D images, texts, 3D VRML files or movies) are provided by a pull-down menu (see Figure 4).

In each of these cases, a new window for opening and editing files pops up. For example, a user might want to open a shell script (in the File-Handling session) and run it on the server (see Figure 5). There is nothing extraordinary in running a script on a remote

Figure 4: The File-Handling session window allows a user to specify the remote directory to monitor and open handling windows for different kinds of files (2D images, text, 3D VRML files and movies).

Figure 5: Clicking on Text files in the pull-down menu shown in Figure 4 brings the text window, through which user can open a local or remote shell or IDL script (depending on the session), edit it and run it on the server.

machine, for which the user has a login. The valuable feature is the fact that the server will monitor the specified directory and bring 2D and 3D images (all files with extension .png, .jpg, .gif, .wrl) to the client automatically. Examples of 2D and 3D images brought to a client are shown in Figures 6 and 7 respectively.

Figure 6: The image window (showing Tesla RF-cavity) in the File-Handling session allows opening remote and local 2D images. User also can use simple markup (rectangles, ovals and texts) to annotate the image.

In case when a user chooses to be in the IDL session, CoReViz starts a remote IDL application and allows user to run IDL scripts and display the results on the client side automatically. In case when IDL application opens or changes X windows, these new windows are brought up as JPEG files to the clients automatically, as described in

section 4.2.

To conclude, the single-user mode of CoReViz presents the users with a convenient way to open local and remote files, run scripts on the server and be updated automatically with 2D and 3D images generated on the server.

4.3. Multi-user mode

In the multi-user mode, the user is presented with the main client window, after authentication and connection to the CoReViz server. (Figure 8). This window lists all the users connected to the server and the all the sessions started by the clients. It provides simple ways to communicate with the people having open sessions: one can broadcast messages or send them to a particular person using a simple chat facility (Figure 8). In addition, the user can create a new session or connect to an existing one (Figure 9). The types of sessions are the same as in the single-user mode: File-Handling and IDL sessions.

Figure 7: VRML file viewed via a built-in VRML viewer allows clients to view and manipulate 3D images. This figure shows isosurfaces of plasma temperature in a tokamak.

Figure 8: The main client window in multi-user mode shows all users connected and sessions started. Simple chat facility allows broadcasting to all people or communicating with a particular person.

Similar to the single-user mode, the client who started the session should enter the path of the directory on the

remote machine where the server will monitor new visualization files. If actions of any of the connected users lead to the generation of visualization files, all the users connected to the session get updated and receive the 2D or 3D images.

Figure 9: The user chooses an existing File-Handling session.

When dealing with files opened locally or brought from the server, users can share them by using the collaboration menu on the text, image, VRML and movie handling windows (see Figure 10). The same menu in the text-handling window allows running the file on the server with the results being shared.

Figure 10: The text-handling window allows user to open local and remote files, share them with all participants of the session, run the scripts on the server and automatically share visualization results as they get generated.

The image-handling window in the multi-user mode allows for collaborative actions: one can distribute the image, or share simple markups and clicks. These when one person wants to attract attention of the team features are helpful in discussion of scientific results to a particular detail of the image (see Figure 11).

Figure 11: In multi-user mode, users may share views, markups and clicks on images.

When dealing with the VRML files (Figure 12), users may choose to share the view as they rotate, pan or zoom in and out in real time, or only when they decide to do this (commit-type sharing).

Figure 12: Interaction with the 3D images can be shared in real time or at a particular moment.

To conclude, the multi-user mode of CoReViz allows opening and sharing of local and remote text, 2D and 3D images, and quick-time movies, running shell scripts and visualization applications (IDL) collaboratively, so that the visualization results are shared automatically in real time or in the commit-type manner. All the file handling frames in both single- and multi-user modes have the capability to open and manipulate multiple files in tabs.

5. Movies generation and sharing

Animation has become increasingly popular for studying dynamic phenomena. Physicists routinely create movies from images to capture a dynamic system at different time steps. To help automate this process and be able to share

the animation, we have implemented the movie generation capability in CoReViz. The movie generator runs on the server and is implemented using the Java Media Framework (JMF 2.1.1e [21]). This generator takes as input a series of JPEG files specified by the client, sorts it alphabetically, so that the series of images are in the proper order, (see Figure 13) and generates a QuickTime movie (.mov), which is stored on the server, in the monitored directory. Using the movie-handling window (Figure 14), users can see the movie and distribute it to all session participants using SCP.

Figure 13: User chooses local or remote .jpg files to generate the movies from.

Figure 14: The movie-handling window allows clients to watch and synchronize the viewing of Quick Time movies.

Once the movie is distributed, all viewers can synchronously watch it, including such controls as restart, pause etc. This controlling information is shared using CORBA calls. One client can assume a leading role by clicking on “Distribute View on Change.” Her manipulations with the movie will then be communicated to other clients by sharing the frame number, so that the team follows her actions and see the same frame of the movie.

6. Conclusions

We have developed an application named CoReViz, which automates typical manual repetitive actions performed by computational scientists in order to visualize remote data without actually bringing the whole data to local

machines. CoReViz allows distributed collaborators to visualize remote data using local and remote scripts, transport and render the visualization objects to multiple users automatically and efficiently.

7. References

- [1] W. Allcock, J. Bresnahan, R. Kettimuthu, M. Link, C. Dumitrescu, I. Raicu, I. Foster, "The Globus Striped GridFTP Framework and Server", to be published in *Proceedings of SuperComputing 2005 (SC05)*, November 2005.
- [2] M. Beck, Y. Ding, T. Moore, J. S. Plank, "Transnet Architecture and Logistical Networking for Distributed Storage", *Workshop on Scalable File Systems and Storage Technologies*, San Francisco, CA, September, 2004.
- [3] V. Bhat, S. Klasky, S. Atchley, M. Beck, D. McCune, M. Parashar, "High Performance Threaded Data Streaming for Large Scale Simulations", *5th IEEE/ACM International Workshop on Grid Computing*, Pittsburgh, PA, November, 2004.
- [4] C. R. Sovinec, T. A. Gianakon, E. D. Held, S. E. Kruger, D. D. Schnack and the NIMROD Team, "NIMROD: a computational laboratory for studying nonlinear fusion magnetohydrodynamics", *Physics of Plasmas*, 10, 1727 (2003).
- [5] <http://w3.pppl.gov/~jchen/>.
- [6] C. Nieter and J. R. Cary, "VORPAL: a versatile plasma simulation code," *Journal of Computational Physics*, Volume 196, Issue 2 (20 May 2004).
- [7] National Energy Research Scientific Computing Center: <http://www.nersc.gov>.
- [8] Hierarchical Data Format (HDF5) home page: <http://hdf.ncsa.uiuc.edu/HDF5/>.
- [9] <http://www.rsinc.com/idl/>.
- [10] <http://software.sci.utah.edu/scirun.html>.
- [11] <http://www.opendx.org>.
- [12] <http://www.ensight.com/products/ensightgold.html>.
- [13] <http://www.sgi.com/products/visualization/van/>.
- [14] <http://www.gnuplot.info>.
- [15] <http://www.imagemagick.org>.
- [16] <http://www.avs.com>.
- [17] <http://www.cs.wustl.edu/~schmidt/TAO.html>.
- [18] <http://www.jacorb.org>.
- [19] <http://www.xfree86.org/4.0.1/Xvfb.1.html>.
- [20] http://www.sra.co.jp/people/nisinaka/Jun4Java/index_en.html.
- [21] <http://java.sun.com/products/java-media/jmf/>.