ElVis: A Portal for Scientific Graphics

Eliot Feibush, Douglas McCune, and Scott Klasky

Abstract—Creating web-based visualizations around legacy fusion plotting codes written in Fortran has been an effective approach for enhancing the graphics and making an interactive system for collaborating scientists. A multi-tier architecture was developed to optimize computing and display resources while making minimal changes to the original Fortran code. The buffering limitation of standard input and output in legacy command line programs was overcome by implementing a pseudo-terminal interface that handles communication on the server side with the command line code.

***Index Terms*—client-server systems, collaboration, graphical user interfaces, web-based scientific visualization.**

I. Introduction

DEVELOPMENT of TRANSP, the transport analysis code suite for tokamak experiments, started over 25 years ago [1]. These programs are still actively used by physicists at several institutions around the world to analyze the data acquired from plasma fusion experiments. The results of an analysis are plotted using Fortran programs that prompt the user with text menus and read command line input. These well established application programs are tailored to extract specific data from an analysis and formulate graphs for plasma fusion research. The suite has significant data handling functions, but its display is limited to only one monochrome Tektronix emulator window as shown in Figure 1.

The goal of this work is to visualize the fusion data and develop a new graphical interface for exploring the data. TRANSP is now available as a compute service so there is the further goal of providing collaborative visualization from a web browser. An analysis may run for several days. Monitoring enables the user to stop an errant run and save computer and human time. These goals have been achieved by developing the ElVis Portal as a multi-tier visualization system. It consists of a client running in a web browser that communicates through a servlet to the legacy plotting programs. The server side of the portal can access the output files of TRANSP runs on the compute cluster. Through a new subroutine interface, the application programs send an XML description of the graph and the data to the Java servlet. The servlet creates a graph object containing the data (not just a static picture) and sends it to the display client. The Java applet has a graphical user interface for exploring the data.

Manuscript submitted June 23, 2006.

Eliot Feibush and Douglas McCune are with the Princeton Plasma Physics Laboratory, Princeton, NJ.

Scott Klasky is with the Oak Ridge National Laboratory, Oak Ridge, TN.

II. Related Work

The ElVis display client is based on the SciVis program [2]. Written in Java, SciVis had a graphical user interface and a socket for receiving data from application programs and collaborating peers. Scientific visualization over the web was described in [3] and articulated as a problem solving environment in [4]. They outline several approaches for dividing the computing and the data between the client and the server. An extensive portal for scientific visualization and volume rendering is described in [5].

III. Display Client

The ElVis display program presents publication quality visualizations in multiple windows as shown in Figure 2. Data from experiments is organized by "shot" of plasma and forms the input to a transport analysis. Graphs from multiple shots and analyses can be displayed in the client. Data from one shot or analysis can be copied and pasted into another to make a direct comparison. A digital crosshair can be interactively positioned for each curve in a graph to show numerical values.

A time indexed graph contains a series of datasets, $f(x_t)$, for the same variable. Displaying the time steps sequentially produces an animation showing the variable's behavior over time. Multiple graphs (with multiple variables) can be animated to visualize several processes simultaneously.

The annotations from the whiteboard, e.g. lines, circles, highlights, are defined in data space coordinates so the annotations zoom and scroll with the view. The data highlighter tool draws a feature along a curve from x_1 to x_2 and is shown in each step of animation. Annotation is saved with the data in a netCDF file. Stored as a URL on the portal, other clients can browse for the file and display it.

The ElVis client implements efficient collaboration similar to SciVis. A graph object sent to the servlet is relayed to each instance of the client that has registered for collaboration. Communicating through the portal enables users to collaborate between machines. As a user interacts with a display, such as zoom, annotate, or animate, only an interaction command is sent. The collaborating applets receive the interaction and apply it which is considerably more efficient than pixel based systems that transmit entire images.

Downloading the applet to the web browser at runtime eliminates installing software, yet always provides the latest version. The visualization subsystem is packaged in an archive file so Java and Jython programs can easily import it. The interactive methods are encapsulated in the classes making the methods available in any application.

IV. Server Side

Applet security restricts file access and limits socket connections so the applet is paired with a servlet that can run programs and access all data on the file system. The servlet runs in the Apache Tomcat container and services client requests forwarded through the firewall by the HTTP server shown in Figure 3. The ElVis application programming interface (API) enables a Fortran or C program to define a *GraphWindow* as a grid of scientific graphs containing datasets. The API produces XML and sends it to the servlet. The XML protocol facilitates extending the API while avoiding version mismatches that could break an input reader. The hierarchical, object-oriented design of the GraphWindow readily maps into an XML description.

Security is enforced by credentials stored on a remote myProxy server. The user just enters name and passphrase in the sign-on panel in the applet. A script conveniently downloads the user's credential to the servlet from the myProxy server instead of requiring the user to manage and upload files. The servlet uses the credential file to run application programs in the Globus Security Infrastructure [6].

V. Pseudo-terminal

The command line interface to TRANSP programs is well established. Many scripts have been written for it. Handling command line standard input and output could not be trivially accomplished with a Linux pipe due to buffering. This limitation was overcome by developing a controller program on the server side that forks a child process of the plotting application. The controller handles text communication with the child through a non-buffering pseudo-terminal and relays it through a socket connection to the servlet. There can be many lines of output for a single line of input. A command that takes a long time to complete may output several groups of lines instead of all lines in one group. This requires accumulating lines in the servlet and the applet repeatedly requesting new data because all client-server exchanges must originate from the applet. The only change to the legacy command line code was appending an end-of-text character to the end of each prompt for user input.

References

[1] R. Hawrlyuk, An empirical approach to tokamak transport, *Physics of Plasmas Close to Thermonuclear Conditions*, Vol. 1, (1980), CEC, Brussels, pp. 19-46.

[2] K. Byeongseob, S. Klasky, Collaborative scientific data visualization, *Concurrency Practice and Experience*, Vol. 9, Issue 1, (Nov. 1997), John Wiley & Sons Ltd, pp. 1249-1259.

[3] K. Brodie, Visfiles: Harnessing the web for scientific visualization, *Computer Graphics*, Vol. 34, Issue 1 (Feb. 2000), ACM, pp. 10-12.

[4] K. Engel, R. Westerman, and T. Ertl, Isosurface extraction techniques for web-based volume visualization, *IEEE Proceedings Visualization '99*, pp. 139-146.

[5] T.J. Jankun-Kelly, *et al*., Deploying web-based visual exploration tools on the grid, *IEEE Computer Graphics and Applications*, Vol. 23, Issue 2, (2003), pp. 40-50.

[6] I. Foster, C. Kesselman, G. Tsudik, S. Tuecke, A security architecture for Computational Grids, *Proc. 5th ACM Conference on Computer and Communications Security*, (1998), p. 83-92.

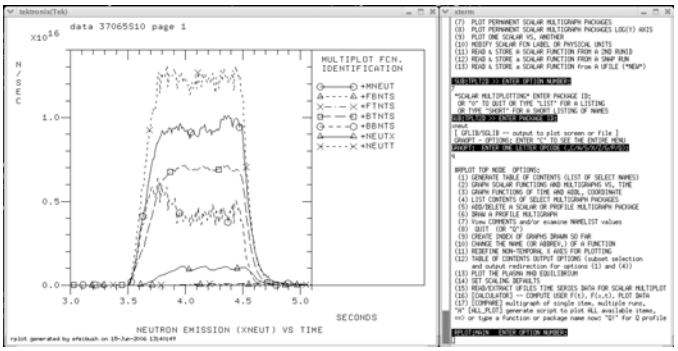

Fig. 1 The legacy plotting programs were written to draw into one monochrome Tektronix window with command line input and text menus.

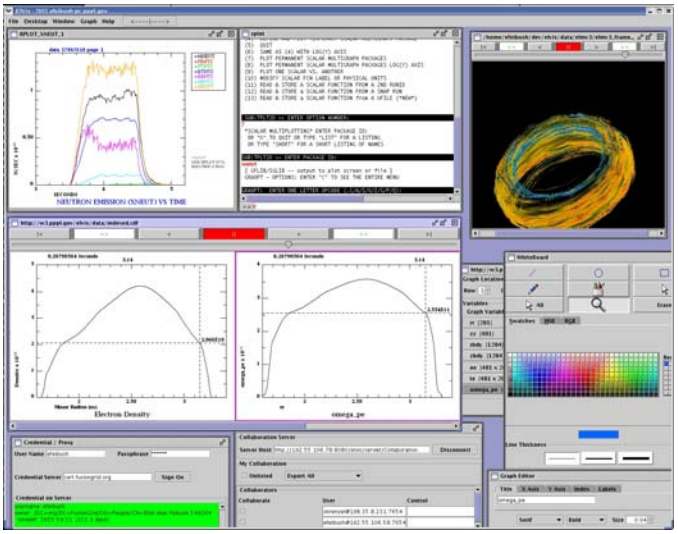

Fig. 2. Graph drawn in the ElVis display client by pseudo-terminal connection to legacy codes on server. Collaborative data exploration from multiple data sources, whiteboard, graph editor, and portal sign-on panels are also in the client.

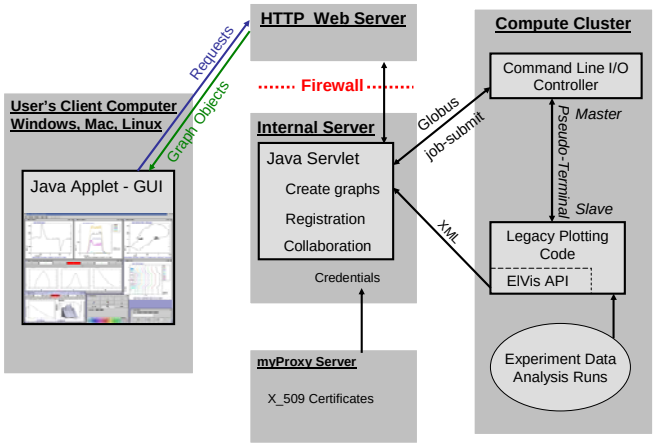

Fig. 3. Multi-tier architecture of the ElVis Portal includes display client, HTTP server, servlet, myProxy server, and legacy applications running on compute servers.