Multi-Tier Graphical Web Service for Simulating Reflectometry in Plasma

Eliot Feibush Gerrit Kramer Ernest Valeo Raffi Nazikian Douglas McCune
Princeton Plasma Physics Laboratory

Abstract

Building a web service around a physics simulation code written in Fortran has been an effective approach for adding graphical input, visualizing the output, and making the simulation available to scientists. A multi-tier system was developed to optimize computing and display resources while making minimal changes to the original Fortran code. The amount of data in the simulation exceeds the memory limit of applets running in browsers. This limitation was overcome by compressing the data on the server before transferring the data to the client. Displaying the results involves blending images at varying resolutions that would also exceed the memory limit of the applet for zoom in viewing transformations. This was solved by mapping the target display region to the source images and transforming only the visible pixels to the blend buffer. A set of reusable scientific graphics classes were developed for upgrading a number of other legacy fusion codes.

Background and Related Work

Reflectometers are diagnostic instruments for fusion experiments. The reflectometer emits radio frequency waves toward the plasma and measures the amplitude of the reflected wave. Turbulence and fluctuation in the plasma can be located by correlating the reflection of different frequency waves. Reflectometers are expensive to build and the number of experimental shots of plasma for acquiring data are limited. Therefore simulating the behavior of a reflectometer is essential before design, fabrication, and deployment. A wave propagation code was written in Fortran. Input and output were file based without any graphical display. The goal was to add a graphical user interface to set up a run, automate visualization of the output, and make it available to physicists at different institutions running various computer platforms.

The idea of an applet sending input to a simulation on a compute server was proposed in **[Fishwick 1996]** soon after the introduction of Java. The server side relied on scripting for processing input forms from the applet. Scientific visualization over the web was described in **[Engel 1999]** and articulated as a problem solving environment in **[Brodlie 2000]**. They outline several approaches for dividing the computing and the data between the client and the server.

Multi-Tier Approach

The graphical user interface, Elfresco, was written in Java to achieve portability of a single version to Windows, Macintosh, and Linux operating systems. Elfresco runs on the scientist's personal computer to take advantage of the tightly coupled graphics and achieves good interactive performance.

Implementing the Java program as an applet enables users to access it through a web browser and then automatically run the latest version without installing application software on their computers. An applet has security restrictions so a Java servlet was developed for controlling the simulation program and accessing files on the server side. The applet sends requests to an HTTP server which forwards them to a Tomcat servlet container inside the firewall. Each run is set up and processed in a protected directory on the server. Then it is listed and managed in the run history window in the applet. For grid portal security each user has an X.509 credential stored on a MyProxy server. The user only enters a password and does not have to maintain any certificate files because the servlet retrieves the credential.

Graphical Techniques

The input plasma contains 2-D cross sections of electron density, temperature, and magnetic field strength. The reflection location for a specific wavelength is calculated on the server and visualized in the applet. This enables the user to efficiently choose wavelengths for studying turbulence at a specific reflection layer in the plasma.

The simulation outputs even larger amounts of 2-D data. The entire input and output datasets are stored in double precision in the run directory on the server. Only the data needed for display is scaled to bytes and sent to the client where it is color-coded. Blending the simulated reflection with the input is very effective for visualizing the reflected waves, particularly when zooming in to the data. There are 6 input images and 3 computed images that can be displayed in any combination. The 2 selected images are blended in an image buffer in the applet. However, zooming the entire pair of images and then clipping to the display area would require an image buffer exceeding the browser's memory limit in addition to being computationally inefficient. This problem was solved by mapping the target display area to both of the selected images and transforming only the visible pixels to the blend buffer. This limits the image buffer to the size of the display area yet enables interactively exploring the data at its full resolution.

The correlation between frequencies is calculated on the server where all the data is accessible. The Java servlet creates a graph object and sends it to the Java applet that has the corresponding class methods for displaying and exploring the graph.

References

[Brodlie 2000] Brodlie, K., Visfiles: Harnessing the Web for Scientific Visualization, In *Computer Graphics*, Vol. 34, Issue 1 (Feb. 2000), ACM, pp. 10-12.

[Engel 1999] Engel, K., Westerman, R., and Ertl, T., Isosurface extraction techniques for Web-based volume visualization, In *Proceedings Visualization '99*, IEEE, pp. 139-146.

[Fishwick 1996] Fishwick, P., Web-Based Simulation: Some Personal Observations, In *Proceedings 1996 Winter Simulation Conference*, ACM Press, pp. 772-779.