Web Based Tools For Visualizing Supercomputer Results

Nancy Rowe, Cray Research, A Silicon Graphics Company, Eagan, Minnesota USA

ABSTRACT: *The World Wide Web has become a familiar interface for many computer users. By developing a web based interface for running animations and displaying results we have created an environment that allows scientists to view results of their data without having to learn a new visualization tool.*

Introduction

The World Wide Web provides a good platform for quickly building a user interface. The Web is fun, exciting and easy to use, and everybody is using it or wants to try it. With the tools currently available exciting web pages can be created in minutes and more tools are constantly being created to make web development easier. Although the Web was designed for use across networks and for interaction with other sites, in the following applications we generally ignore those features and focus on the ability to quickly create an easy to use, dynamic graphical user interface. Because people feel comfortable with the Web, they feel comfortable with an interface built on the Web and this immediate familiarity with the interface makes tools easier to use. We use web pages to build an interface to existing software. The following two cases show how we have used the World Wide Web to make visualization easier.

Case 1 uses the Web to provide a graphical user interface so that non-technical people can give technical demonstrations with easy access to supporting data. Case 2 uses the Web to generate batch graphics for very large data sets that are difficult to view interactively.

Case 1 *Using a web browser as a quick and easy to use interface to tools*

Background

In the Applications Department at Cray Research demonstrations are a frequent and necessary occurrence. The audience for demonstrations varies from local grade school students to American and foreign dignitaries. Generally we are requested to show a broad spectrum of demonstrations. We have experts in a variety of areas, such as crash simulation, environmental science, chemistry, and computational fluid dynamics. Usually only the expert in each field has the knowledge of the current work being done, the ability to easily demonstrate and discuss a computer analysis, and the ability to provide information about related work and to provide informed answers to questions. Organizing a group of knowledgeable people from a variety of areas of expertise is often difficult. The experts are not always readily available at the time of the demonstration, and requiring experts to do demonstrations distracts them from furthering their work. If the experts are doing demonstrations they can't be doing science. By creating a demonstration that shows a variety of scientific disciplines, provides links to frequently asked questions, but can be run by a someone without experience in the discipline being demonstrated we can simplify the demonstration process.

Problem: *Experts required to give too many demonstr ations.*

Solution: *Make it asy for non-expert to give demonstrations*

Goal

Our goal was to create a simple demonstration suite that was easy to use. We set a target of having non-technical people be able to give the demonstrations. The idea was that whoever served as the focal point for making demonstration arrangements would be able to give the demonstration. People could make commitments based on their own schedule rather than having to coordinate the schedules of the experts for each field. We also wanted an interface that was intuitive. The user should be able the give a demonstration by following the instructions in the main menu. For anyone accustomed to using a web browser no practice would be required.

Frequently, questions are asked about specifics of a demonstration. We wanted the demonstrator to be able to provide technical demonstrations and answer questions outside of his area of expertise. Information about the demonstration as well as additional supporting information should be easily available to the demonstrator.

We began this project prior to SGI's acquisition of Cray Research. At that time one of our goals for this demonstration suite was to limit hardware dependencies to Cray machines. We also wanted run part of the demonstration on Cray systems. We wanted the audience to actually be able to see a Cray system being used, rather than just hear someone talk about a Cray system. One stumbling block is that many analyses that require Cray compute power take so long to compute that they are not suitable for demonstration purposes. Our goal then was to inform the audience of how long analyses take on Cray systems and to use the Cray in other ways so that an actual Cray system was still part of the demonstration.

Components

This demonstration used the Web to build a user interface around existing demonstration software. Since animation is a big help in creating an exciting demonstration, our existing system displayed animations of results from analyses that had been run on Cray systems. The animation system ***(Image 1)*** consisted of data stored on a Cray ND40 disk array, display software running on a CRAY J90, a PsiTech frame buffer with a 2k x 2k Sony monitor display device. All these components were connected via an NSC HIPPI switch. The interface to our demonstration software was a initially a command line interface. We replaced the command line interface with a web interface and added links to relevant pages to create our web based demonstration.

Considerations

Large data storage and fast network needed

We had a number of considerations for this project. Animations often consist of large amounts of data which require a large amount of storage space. We increased the storage space requirements by working with high resolution animations. We had several animations with images of 2k x 1k. Each image at this resolution was more than 6 Mbytes of data. One particular animation consisted of 1800 frames. This was more than 11 Gbytes of data for a single animation. This data required a large amount of storage space and high speed networks to show a smooth animation. Cray's ND40 RAID disk array provided the capability to store up to 376 Gbytes of data in a RAID5 configuration. All the equipment we used had HIPPI interface connections for fast data transfer. HIPPI 1600 has a maximum data transfer rate of 200 Mbytes/second. If we transfer 24 2k x 1k frames per second we would need to transfer about 151 M2bytes per second. Twenty-four frames per second would give very smooth animation.

Easy access to additional information

Since the people giving the demonstration would not be experts in the field they were demonstrating, easy access to information about what they were demonstrating was required. We also needed easy access to additional information so that the demonstrators could easily answer questions about what they were demonstrating. The Web provided these features. We created a web page with the basic information about each demonstration and with links to other pertinent information.

Run demonstration on Cray system

We wanted to run the demonstration on the Cray for several reasons. We already had an existing animation display program running on a Cray system. We did not want to have any hardware dependencies other than Cray systems. Visitors are generally much more excited about seeing data coming from a Cray rather than from a video or workstation playback. Our existing animation program highlighted the strength of a Cray network. The existing animation program used a 200 MB/sec HIPPI network to transfer data to a frame buffer.

Use NCSA browser and server

Our decision to run only on a Cray system limited some of our software choices. We were only interested in working with existing web servers and browsers that ran on Cray systems. This led us to choose the NCSA Mosaic browser and NCSA server. Although the Mosaic browser lacked some of the features of Netscape Navigator, it still provided a commonly used web interface on which people felt comfortable working. Our web interface allowed easy integration of animation, text and existing programs. And our web interface allowed easy links to other information.

Animation Software

The software consisted of a C program which used the PsiTech frame buffer libraries to send images to the PsiTech frame buffer. This program was called from a Common Gateway Interface (CGI) script. The interface was an HTML page. Instructions were at the top of the page and users could start and stop animations by clicking on the appropriate image within the page. Summary information was given for each animation. Text within the page could be clicked on to lead the user to additional information for each demonstration.

Experience

Install NCSA browser and server on Cray system

Once we decided to use a web interface, the first step was to get the appropriate web software running on our Cray system. We chose NCSA's HTTPd web server ***(http://hoohoo.ncsa.uiuc.edu)*** and NCSA's MOSAIC web browser ***(http://www.ncsa.uiuc.edu/SDG/Software/XMosaic)*** because they had been ported to the Cray, they were free, they were easy to obtain via anonymous ftp, and we were familiar with them from working with the workstation versions. When we began the project there were fewer web building tools than there are today. Most of our initial development work was done using a screen editor. Even with this limitation we found it easy to create an attractive demo interface. Currently there are many more web development tools available that would make the job easier.

Internal Use

This demo was only intended for internal use, so we were not limited by some of the restrictions that other web developers faced. Because we were dictating that Mosaic be used for the Web browser we were unconcerned about how the pages appeared using other web browsers, however this also limited us to using only features supported by the Mosaic browser, and we were unable to use features such as frames and tables.

Simple HTML pages

The demo pages were simple. They consisted of a main page with an instruction section. ***(Image 2)*** The page contained images and text for a variety of demonstrations. The user was instructed to click on a image to start the animation. By clicking on an image the user would start a process on the Cray system that would send images from the ND-40 disk array to the PsiTech frame buffer and Sony monitor as an animation. This process was completely invisible to the user. The user clicked an image and then saw the animation on the Sony monitor. Additional pages give more detail for each demonstarion and provide links to sites with related information.

Tools

Initially when we created our web pages we used the visual editor, vi, since we did not have access to any other web building tools. With Silicon Graphic's acquisition of Cray Research we have access to many more visualizaiton tools. Recently we have worked with SGI's CosmoCreate and Adobe Photoshop. CosmoCreate is SGI's visual HTML editor. With CosmoCreate we were able to very rapidly create attractive HTML pages. CosmoCreate let us design pages, add links, add images and test pages. With Adobe Photoshop we were able to modify and enhance images. These two tools allowed us to quickly prototype a web user interface.

Running the browser and server

We built a special web server for this demonstration. Web servers are not generally used on Cray systems. We ran the server as an application. A README file instructed the demonstrator to run a prebuilt script to start the server before the demonstration. We chose this method because at our site the Cray system is rebooted and reconfigured on a regular basis. By running the server as an application rather than as part of the Cray system we had complete control over the software and we were not dependent on the web server being brought up as part of the kernel.

<u>Case 2</u> *Using the Web to create a viewer for analyses*

Background

Cray systems are very often used to analyze large data sets. This often generates large results files. As the results files get larger, it becomes more difficult to do interactive visualization of results. Computer users with large results files would like to easily visualize the results. By using the Web to add a simple graphics script to an analysis job before it is run or using the Web to submit the graphics script after the job is completed, the desired images and animations are generated for later playback. The playback of the image and animations can also be done using a web interface. The worldwide availability of the Web allows results to easily be shared to with researchers in other geographic locations. The wide spread acceptance of the Web allows this to be done without the difficulty of installing and supporting additional software.

Problem: *Data too large to be view interactively*

Solution: *Have batch job generate desired visualization*

Goal

The initial goal of this demonstration was to provide a simple interface to allow users to create visualizations of results from large data sets. The Web interface was designed to allow the user to take an existing input script for the engineering analysis code Livermore Software Technology Corporation's LS-DYNA3D, edit it using the web interface, select desired visualization output and submit the job to a Cray or SGI system after selecting computation parameters.

There are a number of advantages to building software on the Web in this way. The Web is a familiar platform. The familiarity of the Web is beneficial to both the developers and the users. Developers are accustomed to working with the Web and can build applications based on their experience using web software. Users are more comfortable using an interface they are familiar with. Our goal for this type of software is that fewer experts will be required to do analysis. The web interface is generally easier to use than a typical application interface. Scripts can be generated to provide a template for analysis. One expert could oversee the work of several people. Each person would run their analyses using a web interface and a template script to which minor changes could be made. More analyses could be run.

This project is a 'work in progress'. We are currently exploring the concept of using the Web to build tools. This is an open ended effort. The goal of the project is not to create a product, but to research the concept of the Web as a platform for more advance visualization tools, and to prototype visualization tools on the Web so that the concept may be more easily critiqued. We currently have many questions, some of which we will not be able to answer until we see what happens with the Web.

Components

This web application interfaced to application software running on Cray computer systems.

This application is designed as a modular system and components can be easily changed. There are four main components in our system: Web interface for designing visualization scripts; Analysis code; Visualization code; and Results viewer. In our system we used LS-DYNA for the analysis code and wrote software for the web interface, the visualization scripting software, and the results viewer. We could easily replace LS-DYNA with another analysis code, or we could use software from Alias/Wavefront to do the visualization. The modularity of the system offers many possibilities.

System

The Web software consisted of NSCA web server and Netscape Navigator browser. The interface was written using a combination of HTML, JavaScript, and C. We ran the browser on an SGI Indigo 2 workstation. Images were displayed using SGI's ipaste and movies were displayed using SGI's movieplayer. On the Cray system we used NQS to submit scripts to LS-DYNA. We developed C code on the Cray system to generate the images and movies.

Experience

The main page for the Web Visualizer ***(Image 3)*** consists of 4 selections: Review Dyna Deck, Submit Dyna Deck, Batch Graphics, Analysis Results.

Review Dyna Deck allows the user to load in an existing Dyna input file and review and edit the parameters.

Submit DYNA Deck is basically a graphical user interface to the Network Queuing System which allows you to set up the parameters for NQS and submit the job to the Cray system.

The Batch Graphics Page ***(Image 4)*** allows the user to specify what images and movies he would like to see. The user can choose from a palate of graphics functions such as isosurface and slice plane. The user clicks on the desired graphics functions to build up each image or animation. Once the script has been built ***(Image 5)*** the user clicks on the "Submit Script" button and a CGI script initiates the application software on the specified system.

The Analysis Results Page ***(Image 6)*** shows the visualization results. This page dynamically updates as new images and movies are generated. ***(Image 7)*** Small images in the page represent the images or movies that have been generated. The user clicks on one of the small images on the page to see that actual image or movie.

The network scope of this application was generally kept to within Cray Research. Web security issues make it difficult to submit jobs to or from remote sites. In order to easily use this application we needed to be able to create software that was run by the server such as CGI scripts.

Even though the actual network to generate analysis results was local, sharing the results with remote sites is possible. The Analysis Results Page is available to remote users via the Web. The availability can be controlled so that results may be kept accessible only to those on the internal network or the page may be moved to a different location to allow accessibility from other locations.

Questions and Issues

The following is a sampling of some questions and issues that have arisen during our work on the Web.

How do we handle the security issues? Many people are not yet comfortable with security available on the Web. Until the comfort level is raised it will be difficult to get these people to use Web based tools to visualize their data. Currently there are a few options to providing more secure web pages, such as setting up password checks or using authorization files, but these efforts to provide a more secure environment make more work for the developer and often create a less user friendly interface. And even adding these security features people still feel unsure about the security of their data.

As data sizes increase how do we ensure that the data is located in the appropriate place? Transferring large data files across a network can be difficult and space can be a limitation. On the Web if a file is not located in the correct directory, it may be impossible for the user to access the file. Sorting out how to be sure that data is in the correct location is an issue for this type of software.

Conclusion

The Web is a good platform for building simple visualization interfaces. People are comfortable using the Web. With the many web tools available building an attractive user interface is easy. Software installation and maintenance is simplified because so many sites support web browsers and servers. Because web software is supported on most major computer platforms, available hardware issues are simplified. Networking facilities in the Web allow data to be shared with others who have web access. By providing and easy to use interface to visualizaiton we reduce the expertise bottleneck.

Although the Web is a good platform for simple interfaces, people creating more complex interfaces should consider several factors to determine if the Web will be an appropriate interface platform. One thing to consider is that on the Web maintaining data security can be an issue. Very often the user is not running the application, the server is running the application. This means that the server may require the data to be world readable. Anyone with access to the server would have access to the data. Another consideration is that building a more complex interface, the developer will likely want to be able to work with the web server to generate CGI scripts. At some sites it may be difficult to get access to the web server.