Designing a Distributed Scientific Visualization Tool

Lisa Vander Sluis

Technology Applications, Inc.

Abstract

Technology Applications, Inc. (TAI) provides supercomputing services for the Naval Underwater Systems Center (NUSC) in New London, Connecticut and Newport, Rhode Island. Many NUSC engineers and scientists use the CRAY X-MP/28 to accomplish their work in such diverse areas as acoustic modeling, finite element analysis, and computational fluid dynamics. This paper focuses on the benefits of using a distributed scientific visualization tool in the field of acoustic modeling.

1.0 Design Requirements

Mark Peloquin and Rick Dumont, NUSC engineers, were working on the problem of how an acoustic wave propagates through a hydrophone. They would use acoustic modeling to compare prototype performance with theoretical expectations. They were using a Sun workstation to do their calculations but required a more effective tool for their development work. Ideally, they wanted this tool to allow them to interactively input and modify parameters on a workstation, calculate the compute-intensive modules on the CRAY, and immediately display the results on a graphics workstation. In addition, they wanted the use of a remote compute server to be transparent. They also needed the flexibility to use the power of the CRAY while at NUSC, but be able to run their code entirely on a Sun while on a sea test when only a Sun is available. Furthermore, to obtain funding and approval of their project, they needed to produce high quality presentations.

1.1 Solution Paradigm

With the engineering requirements in mind, a three-part solution was designed. First, a user-friendly interface was developed under SunView. Next, a Remote Procedure Call (RPC) was used for transparent data transfer between a CRAY X-MP/28 and Sun 4 workstation. Finally, PV-WAVE, a high-level graphics package by Precision Visuals, was used to visualize the results. The architecture of the distributed solution is shown on the next page.

2.0 SunView

SunView was chosen for the user interface over the X Window System because the engineers were accustomed to it, and because PV-WAVE did not have a driver for X at the time the program was designed. This user interface facilitates the input and modification of data parameters and allows nearly-transparent selection of the compute server by choosing from on-screen buttons indicating the available machines.

2.1 Considerations in Designing a User-friendly Interface

To ease acceptance of the new interface, it was important to meet with the users to learn how they were doing their work. The engineers originally had only a command line data input interface. This meant that the user-friendly interface had to be designed from scratch. At the same time, they were continuing development of their algorithm. This made it more difficult to finalize the design of the graphical user interface. However, it was important to provide a preliminary version of the graphical interface as soon as possible to make the engineers' development work easier.

The initial graphical interface included non-operational on-screen buttons that could be used for additional input parameters later. In spite of this, it still was necessary to redesign the user-interface screen once during the project. This redesign placed buttons representing parameters for a single entity near each other for ease of data input. Also, default values for the input parameters were added so that only those needing change had to be entered. This increased the speed and accuracy of data entry. The users were allowed to name the input variables to reduce their learning curve.

Having all the input buttons displayed on the screen simultaneously kept the user's memorization requirements low; but all these buttons did hog the screen. Because this was an interactive program in which the engineers could track the ongoing simulation, two important functions were needed:
(1) A button which would print out the screen to save the data input of successful runs.

(2) A button to save the numerical results so that the data could be used to generate another PV-WAVE graph without rerunning the program.

Button (1) was modified later to print the input data to a file rather than as a screen dump.

3.0 Remote Procedure Call

An RPC gives a programmer a lot of flexibility in optimally designing a distributed program. With an RPC, the programmer may call a subroutine on either a remote machine or the local machine. Sun’s RPC uses eXternal Data Representation (XDR) to allow portability between unlike machines, and data transfers can be transparent to the end-user.

Some features of the workstation-CRAY paradigm are:
- The user-friendly interface
- Visualization of the results
- The CRAY is beneficial for:
  - Compute-intensive calculations
  - Fast I/O and data transfers

3.1 Designing the RPC Configuration

First, an optimal client/server distribution of the original program, which existed wholly on a Sun, was designed. Some design decisions were innate to the goal of distribution such as: using the Sun for data input and porting compute-intensive subroutines to the CRAY. A major goal was to keep the distributed version similar to the engineers’ original program structure.

The program was built around a single main loop that called numerous subroutines to perform the acoustic calculations and to display the results. The most difficult decision was how to modify this main loop. In the end, the main loop was kept on the client (Sun) and an RPC was used to call the compute-intensive subroutines which were placed on the server (CRAY). This software architecture is illustrated in Figure 1, above. This way, the results are plotted each time they are calculated. By plotting the results after each calculation, the program was made more interactive and allows the engineers to track their computations as they progress rather than waiting to display the results after all the computations are done. This saves time and money that would be lost if the results were displayed only after the program finishes running. If there is a problem with the calculations, the users can stop the computing process and modify their input parameters immediately.

3.2 Using RPC

To use RPC, the variables to be sent to the server and the parameters to be returned to the client must be defined in XDR format, a standard format for inter-machine communication. On each machine, the variables first are incorporated in a user-defined structure, then the RPCGEN compiler is
used to create the C header files and XDR routines needed for the communication.

There are three layers of RPC from which to choose. The differences among layers are in ease of programming and in inherent restrictions. The RPC documentation recommends that most applications begin by using the second layer, so the original program design did. However, there are several instances in which the lowest level should be used. These are:

- For data transfers of 8K bytes or more
- To perform authentication on either the client or server
- To allocate or deallocate memory while serializing or deserializing data with XDR routines
- To have more control over the timeout period of calls from the client to the server

The engineers later realized that they needed to send more than 8K bytes of data. This meant that the default timeout had to be increased. These factors required a modification to use the lowest level of RPC.

RPC has been incorporated into the new release of PV-WAVE (Version 3.0). It now has call_unix, unix_list, and unix_reply procedures that are essentially RPC procedures allowing PV-WAVE to communicate with C external routines. However, there is no version of PV-WAVE for the CRAY and, according to Precision Visuals, it will not be ported, so this new feature will not help access the CRAY using RPC. Using the new version on the Sun would require keeping different versions of the distributed code. Therefore, we still must use our own RPC communication code to combine RPC with PV-WAVE.

4.0 PV-WAVE

Originally, the engineers used PHIGS (Programmer Hierarchical Interactive Graphics System) to visualize their results. However, since they are engineers, not programmers, they were overwhelmed by the complexities of PHIGS. Because PV-WAVE expression syntax resembles FORTRAN or BASIC, it is easier for an engineer to learn since most have had some experience with one or both of these computer languages. Therefore, they purchased PV-WAVE.

4.1 Visualizing Data with PV-WAVE

PV-WAVE consists of three user levels, the command level, the procedure level, and the application level. The command level is used to enter commands that immediately analyze and display the data. At the procedure level, the user can create standard procedures for performing repetitive tasks. The application level can be used to create user-interface menus to select options such as printing a hardcopy of the plot.

The application level's user-interface menus are currently only available on a Sun under SunView. PV-WAVE plans to port them to X/Motif. Since the engineers are interested in converting to the X Window System, they will wait to use this feature until it is available under X.

To aid in software management, the PV-WAVE commands are placed in files which are then called from the FORTRAN program on the workstation. These files contain commands which display certain characteristics of the engineers' results in several different ways. First, they use the 'show3' procedure to overlay an image, surface mesh, and contour plot. The image plot is a top-down image showing the propagation of a wave. The surface mesh and contour plots are different plots of the same results overlaid on the image plot.

They also may display a 3-D surface plot, using a PV-WAVE surface command that first builds a wire mesh. Data points then are projected onto a 2-D surface after rotating about the Z then X axis. Hidden lines are removed, and the shaded surface is computed using a virtual light source. This 3-D surface plot displays the same results as the overlaid image plot, but it is more useful to them because it gives a better indication of what is actually occurring, especially with a zoom feature for detailed inspection of a specific area in the 3D-plot.

4.2 PV-WAVE Features

The shading methods available in PV-WAVE are constant or Gouraud shading. Constant shading is faster because one shading value is computed for the entire polygon, whereas the Gouraud shading of a surface will be computed at each elevation point and then interpolated for the other points. However, Gouraud gives a better shading of the surface, and it is the default.

To smooth the contours, the 'rebin' procedure can decrease sampling in x and y by a specified factor. Also, by decreasing the number of data values, the drawing time is decreased because less computation time is required.

The user can change the color scale of the plots. In this case, the engineers usually use a shade of red, because it gives much better contrast with the black background. They were concerned that, as their amplitude values varied, the colors in their plots also varied. This made it difficult to compare plots. They were using the 'tvsc' procedure which automatically scales the image into the intensity range of the display device. By using the 'tv' procedure instead, the image will be plotted without alteration. If they want to set their own minimum and maximum values, they may use the 'bytscl' procedure. In addition, the combination of the 'tv' and 'bytscl' procedures is faster because only one pass through the data, instead of four, is needed to do the range truncation and scaling.

385
To get a hardcopy of their visualization, they use Sun's 'acquire' command which does a screen dump to their Tektronix printer. This has 72 dpi resolution. They would like to get a Postscript color printer with 300 dpi resolution so their axis labels would be readable. After printing out transparencies of their plots on the Tektronix printer, they make presentation slides.

4.3 Future Enhancements

For additional enhancements, the engineers would like to change rotation to view the 3-D surface plot from different perspectives. An animation of the rotation would make an interesting presentation. Animation can be done in just a few commands. The procedure would be to display the surface plot, put in 3-D structure, take another surface of it, build up a series of frames, then play them all back.

For I/O optimization on the Sun, the engineers could modify 'for loops' in PV-WAVE procedures to use arrays. They now use two files for x and y array values, (i.e. the x values are all stored in one file and the y values are stored in the other file). They could combine the x and y values into one file. Also, tests and filtering of data could be streamlined by replacing nested loops with array syntax. Similar to the CRAY's vectorization, PV-WAVE's array operations can process several data values at once thereby decreasing the computational time required.

Originally PV-WAVE required input data in ASCII. Now the user also can use binary representation. In addition, PV-WAVE now has XDR. Unformatted I/O is faster than formatted I/O, but it is not portable between machines with different architectures. On the other hand, XDR falls between formatted I/O and unformatted, but has the advantage of portability. Since PV-WAVE is not on the CRAY, its XDR routine cannot be used on the CRAY. However, the large ASCII data file on the CRAY can be written in XDR format using a C program. When NFS is installed on the CRAY at our site, the engineers can then use PV-WAVE's XDR routine on the Sun to read this CRAY file into their PV-WAVE plotting procedure. This will avoid having to use an 'rcp' to move the ASCII data file from the CRAY to the Sun every time the program is executed.

5.0 Conclusion

This computing paradigm has given the engineers a much more efficient tool for their computational research. The user-friendly interface has increased their productivity, and PV-WAVE has allowed them to visualize their results quickly and clearly. The use of PV-WAVE also allows them to produce high-quality graphical output for presentations. In addition, by using RPC, they have gained flexibility in their choice of the machine or which to run the compute-intensive subroutines. This allows them to use the power of a CRAY when at NUSC, while retaining the ability to use their software during sea tests when only a Sun is available. Although this tool was written specifically for a single research project, the usefulness of this paradigm has become apparent to others and is being requested as a general-purpose tool for other engineering projects at NUSC.

Acknowledgments:

This work was performed for the Naval Underwater Systems Center under Contract No. N66604-88-C-0033.

I would like to thank Mark Peloquin and Rick Dumont for their willingness to try new computing paradigms.

Thanks to Dave Finley for helping me in preparing this manuscript.

References