The WebPDELab Server: A Problem Solving Environment for Partial Differential Equations Applications

Elias N. Houstis, Ann C. Catlin, Nitesh Dhanjani, and John R. Rice
Department of Computer Sciences
Purdue University
West Lafayette, IN 47906, USA.

January, 2000

Abstract

WebPDELab is a World Wide Web server that allows users to define, solve and analyze partial differential equation (PDE) problems using a comprehensive graphical user interface from any Java-enabled browser on a wide variety of platforms. The WebPDELab server is currently supported by a 16 CPU Intel cluster, which allows users to solve PDE problems sequentially or in parallel on the supporting host cluster. WebPDELab is the PELLPACK [Houstis et al., 1998] problem solving environment implemented as an Internet-based client-server application. It provides access to a library of PDE solvers and an interactive graphical interface that support the pre-processing and post-processing phases of sequential and parallel PDE computing. The PELLPACK software is implemented as a system of X windows programs and libraries, compiled on an i86pc SunOS 5.6 machine. WebPDELab displays the interface of the PELLPACK software within a Java-capable browser using the Virtual Network Computing (VNC) [Richardson et al., 1998] remote display system.

1 The WebPDELab Server

WebPDELab is a World Wide Web server that provides access to PELLPACK, a sophisticated problem solving environment for Partial Differential Equation (PDE) problems. Users can connect to the WebPDELab site at `http://webpellpack.cs.purdue.edu` with any Java-enabled brower for information, demonstrations, cases studies and PDE problem solving service. The scenario illustrated in Figure 1 shows how a user on the Internet accesses WebPDELab services.

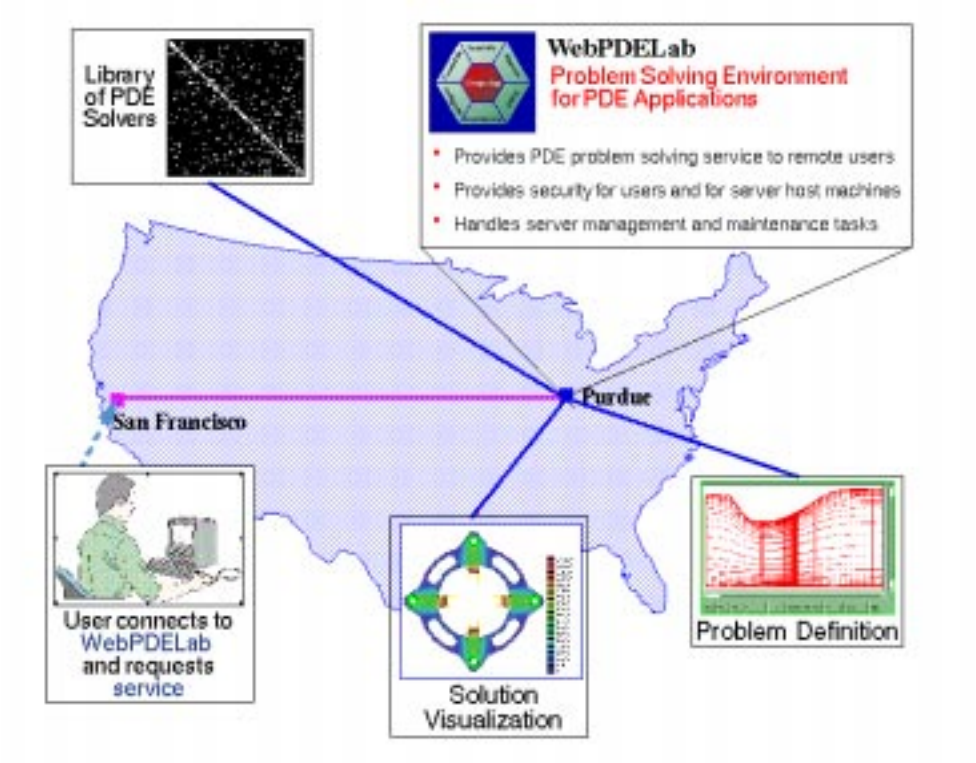

Figure 1: View of the WebPDELab system operating over the Net.

A new PELLPACK session is initiated for each user that connects to the WebPDELab server, and a unique identification and private file space for the session are created. The file space is available until the user disconnects from the service, at which time the session is terminated and the user's files are deleted. Users may download files generated by PELLPACK to their own machines before terminating the session, and they may upload files to WebPDELab at the start of subsequent server sessions. When the server invokes the PELLPACK system software, the entire PDE problem solving environment described in [Houstis et al., 1998] is presented to the user. This environment is described only briefly in Section 1.1. For a detailed decription of the functionality and operation of the PELLPACK software, all or part of the User Guide (400 pages) [Catlin et al., 2000] can be downloaded from the web site. PELLPACK is a comprehensive system for modeling physical objects based on PDEs, and has been used by hundreds of students and faculty both inside and outside of Purdue University for solving problems in physics (liquid crystal droplets, proton flux propogation), thermal field analysis, fluid dynamics, semiconductors, geophysical research, electomagnetic field analysis, thermo-elasticity, structural analysis, and other scientific and engineering applications. PELLPACK has a user friendly interface, and even first time users can solve interesting problems by

following the fully documented, step-by-step descriptions of the problem-solving process presented in `Getting Started` at the WebPDELab site.

1.1 The PELLPACK Problem Solving Environment

WebPDELab is a Internet-based client-server implementation of the PELLPACK software. PELLPACK is a system that allows users to specify and solve PDE problems on a target computational platform and to visualize the solution. PELLPACK provides a graphical user interface for defining the PDE model and selecting solution methods (see Figure 2), and is supported by the MAXIMA symbolic system and well-known numerical libraries. The graphical interface is implemented on top of a very high level PDE language. Users can specify their PDE problem and its solution visually using the graphical interface or textually using the "natural" language. PELLPACK has incorporated over 100 solvers of various types which cover all the common PDE applications in 2 and 3 dimensions.

In the PELLPACK system, a problem is represented by the PDE objects involved: PDE model or equations, domain, conditions on the domain boundary, solution methods, and output requirements. The PELLPACK interface consists of many graphical tools and supporting software to assist users in building a problem definition. A textual specification of these objects comprise PELLPACK's natural PDE language, and the language representation of each object is generated by the object editors/tools. The language definition of a user's problem (the .e file) is automatically passed to PELLPACK's language processor, which translates the problem into a Fortran driver program, and then compiles and links it with numerical libraries containing the user-specified solver methods. Sequential or parallel program execution is a one-step process; the program is executed on one or more machines in the supporting i86pc host cluster. Problem solutions are passed to the PELLPACK visualization system for solution display and analysis.

1.2 The WebPDELab Interface

The WebPDELab server is accessed from the WebPDELab web site. This web site is an instructional source for anyone interested in solving PDE applications. It provides information about PDE problem solving in general, and about the process of solving PDE problems with PELLPACK in particular. A collection of fully documented case studies is available at the site (Figure 3), presenting step-by-step solutions of common PDE applications (flow, heat transfer, electo-magnetism, conduction), with every user action and PELLPACK result described with images and detailed text.

Users who request the PDELab problem solving service must first register with WebPDELab (Figure 4). After the user registration information is validated and the server connects to a host machine, WebPDELab presents a framed HTML page, with a control panel in the top frame (Figure 5) consisting of four buttons: `Upload Files`, `Download Files`, `Start Server` and `Exit`

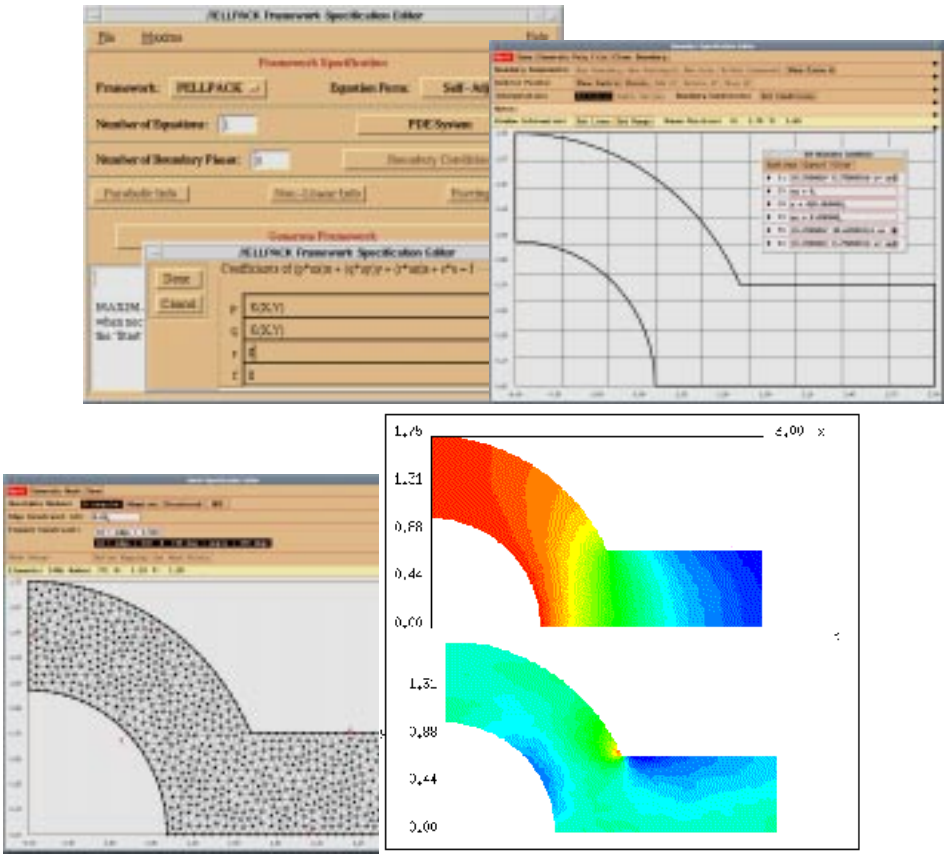

Figure 2: Graphical tools available in the PELLPACK problem solving environment.

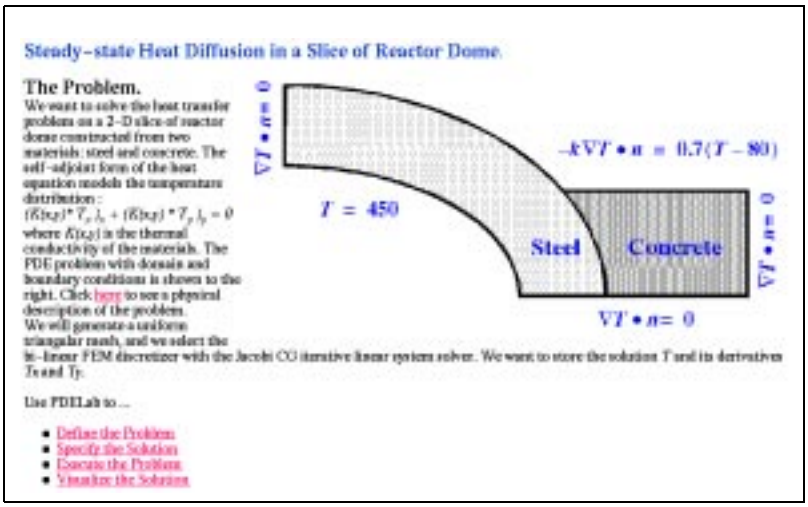

Figure 3: Sample case study from `Getting Started` at the web site.

Figure 4: WebPDELab registration.

`Server`. The bottom frame contains the user identification number, host connection information, and instructions for using the buttons of the control panel in the top frame. At this point, the WebPDELab server has already created the user's directory space, so users can upload files to their directory using the Upload button. Generally users upload PELLPACK problem definition files from previous WebPDELab sessions, such as .e files, mesh files and solution files. Users can upload up to 24 files to their assigned directory space, and files may no longer be uploaded once a user clicks on the Start Server button.

The Download button returns a listing of the user's directory contents. Files in this directory can be viewed or dowloaded from the listing, but since users' directories are password protected, no other directories can be viewed or entered.

Figure 5: WebPDELab server with control panel in the top frame and panel instructions and connection information in the bottom frame.

The Download button is available throughout the user's PELLPACK session. Users should look here frequently during the session to check on PELLPACK generated problem, solution and trace files. The Start Server button invokes the password protected PELLPACK software. After the password is entered and verified (Figure 6), the top level window of the PELLPACK system appears in the bottom frame of the browser window as shown in Figure 7. A collection of sample problems has been placed in the user's directory, so users can load an example into the PELLACK session or begin their own problem definition. The PELLPACK session in Figure 8 is in the bottom frame of the WebPDELab server. The buttons of the control panel are still available in the top frame, but only the Download and Exit Server buttons are enabled. The Upload and Start Server buttons remain disabled while the PELLPACK software is running in the bottom frame.

During the PELLPACK session, WebPDELab is passing the display of the remotely executing PELLPACK environment to the users browser window. The graphical interface displayed on the user's screen belongs to PELLPACK and is not described in this paper. When users click on Exit Server, the PELLPACK session is terminated and the user's directory is removed.

WebPDELab traces all user activities from the start of the server session until its termination. Users files are secure from other users, but WebPDELab 'looks at' the contents of every file uploaded to WebPDELab or created by the user from within the PELLPACK system. WebPDELab protective mechanisms implemented for the security of the WebPDELab server and host cluster are discussed in Section1.4

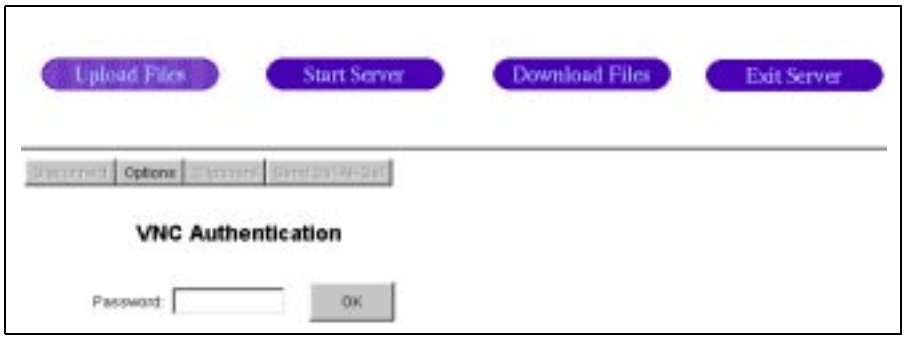

Figure 6: Password entry for the PELLPACK system, showing the control panel in the top frame.

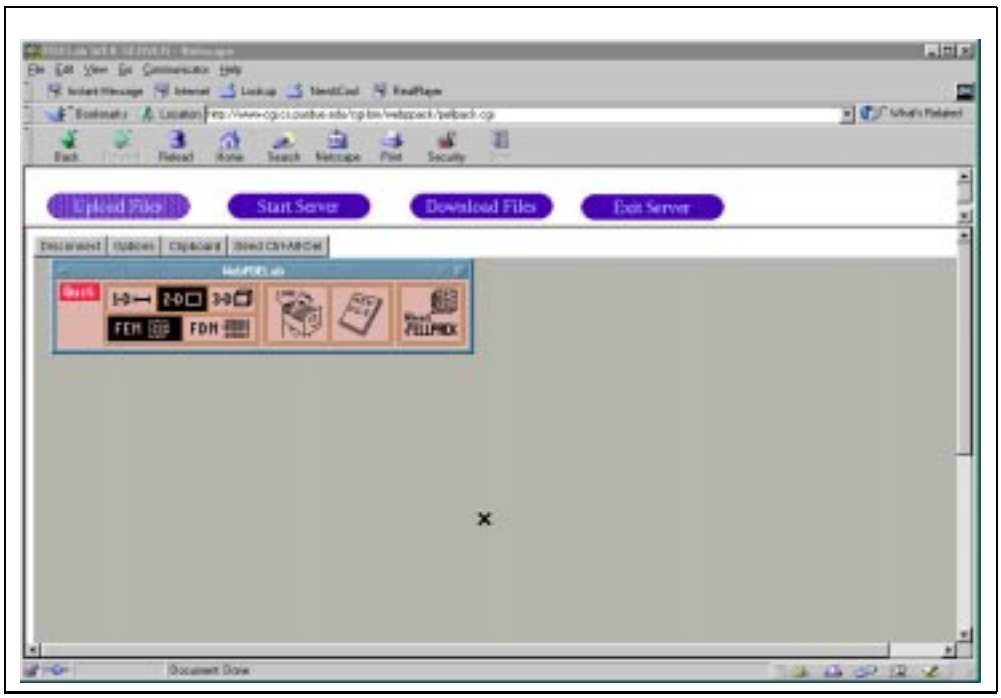

Figure 7: The PELLPACK top level window appears in the bottom frame of the WebPDELab browser window. It is ready for user interaction.

1.3 WebPDELab Implementation

WebPDELab is the PELLPACK problem solving environment implemented as a web server using Virtual Network Computing (VNC) [Richardson et al., 1998]. VNC is a remote display system which allows users to view a computing "desktop" environment from anywhere on the Internet using a wide variety of machine

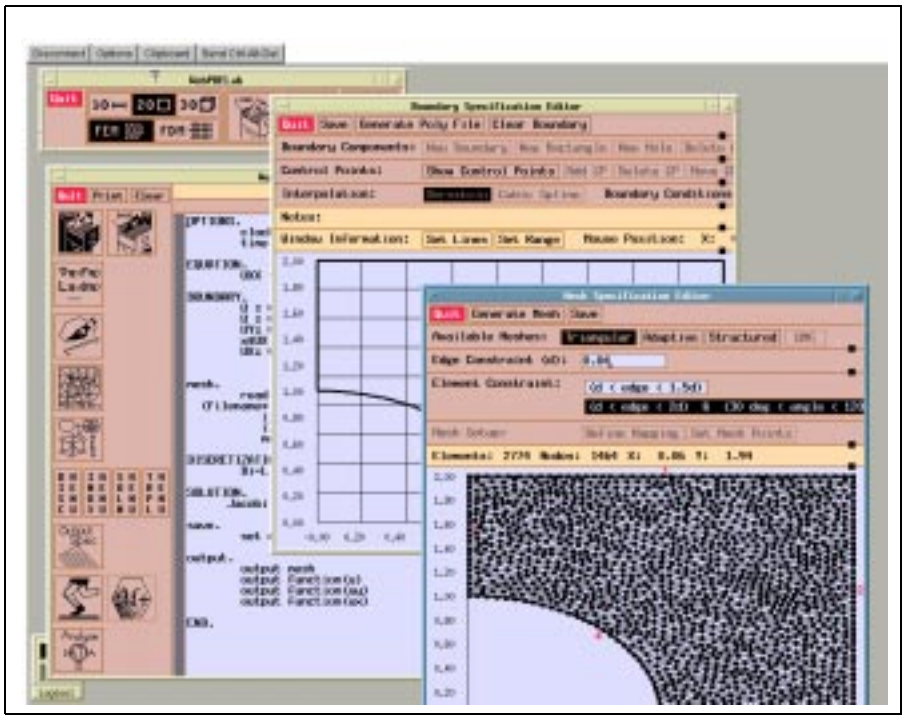

Figure 8: PELLPACK session running inside the WebPDELab browser window.

architectures. VNC consists of a server which runs the applications and generates the display, a viewer which draws the display on the client screen, and a TCP/IP connection between them. The server is started on the machine where the desktop resides, after which any number of viewers can then be started and connected to the server. This allows the client user to access the applications, data, and entire desktop environment provided by the server. The viewer is a small, sharable, platform-independent, and stateless system which runs on the client machine.

In the WebPDELab implementation, a new VNC Unix server is started for each user who accesses the WebPELab web server from a Java-enabled browser (see Figure 9). The VNC Java viewer is started from the user's browser, allowing the user to display and interact with the PELLPACK environment, which consists of X windows programs and libraries compiled and running on the i86pc SunOS 5.6 host machines. Within this framework, any user world-wide who is connected to the Internet and has access to a Java-capable browser can run WebPDELab.

The WebPDELab *manager* is the collection of CGI scripts (Common Gateway Interface protocol for browser to server communication) which control all user activity once the PDELab Server button at the WebPDELab web site is pressed. When a user accesses the server, the manager collects information

on all currently running VNC servers from the host machines. The manager then asks the potential user to enter registration information, including a valid e-mail address. After the email address is validated, a unique user id is generated for the new user, and a log file is set up to track registration information, user access/exit times, and user activities while running the PELLPACK software. The host machine with lightest traffic is selected by the manager for running the VNC server and subsequently the PELLPACK software. A protective client-server application is used to launch the VNC server, so that users are never logged in to any machine in the host cluster. The VNC server startup invokes the PELLPACK system, and the manager creates the user directory, sends the control panel to the user, and monitors the user's interaction with the control panel buttons (Upload Files, Download Files, Start Server and Exit Server).

`Upload Files` is implemented using copyrighted public domain code at `http://stein.cshl.org/WWW/software/CGI` (Lincoln D. Stein, 1998). The code has been modified to operate with the WebPDELab/VNC user directory privacy restrictions. The `Download Files` button is implemented as a standard link to the user's file space, but additional password security protects a user's assigned directory from all other users on the Internet. `Start Server` connects the VNC client user to the VNC server which has been instantiated for the caller on the selected host for a specific VNC server.

After control has passed to the VNC client, the manager waits for a VNC disconnect or an `Exit Server` button click. When signalled to start exit processing, the manager saves the trace of user activities to the log data base, kills the VNC server, and removes the user's directory. The manager also checks all executing VNC servers periodically for sessions running longer than 10 hours, and these sessions are terminated. When the manager has finished exit processing, control is returned to the WebPDELab home page.

1.4 WebPDELab Security Issues

All internet based services must be concerned with security issues and strive to protect their network and host environment from unauthorized access. WebPDELab implements measures to provide such a secure environment by enforcing common rules of best practices which are used to secure Unix machines, taking advantage of the strength and flexibility of the Unix operating system. WebPDELab maintains several levels of security provided by the operating system, the WebPDELab and VNC servers, and protective language processing software built on top of the PELLPACK system. These security measures are described in this section.

When a user logs into the WebPDELab server, a CGI script is executed which generates a unique UID (user identification) for that user and requests one of the cluster host machines to invoke a VNC X-server. The WebPDELab CGI scripts reside on an isolated machine dedicated to serving CGI requests. This machine does not have any NFS-mounted disks, therefore an attacker attempting to take advantage of vulnerable CGI scripts is locked into the cgi-bin directory

and cannot gain access to any other machines or disks. All parameters passed to WebPDELab CGI scripts are scanned to ensure they contain precisely the expected values (argument number,length and contents), else the request is terminated,

The cluster machines listen on a fixed port for startup requests from the CGI machine. In case an attempt is made to connect to this port which does not originate from the CGI host, the connection is immediately terminated. All cluster machines run a daemon which listens for socket connections on a specified port and spawn a child process to serve the request, while the parent continues to listen for other connections so that requests can be served simultaneously. A client program is invoked by the CGI script to contact the cluster machine and request that a new VNC X-server be launched. The client may only specify the VNC X-server startup parameters, since the launching of the VNC X-server binary is hard-coded in the configuration file of the daemon serving requests originating from the CGI host. The VNC server itself is protected by a challenge-response password scheme.

The cluster machines run the VNC X-server as owned by a dedicated account whose root directory is the account's home directory (using the Unix maintenance *chroot* command). All the required binaries are located in this directory. If a user discovers vulnerabilities in one of the cluster machines, the user is locked into the home directory of the account, and is unable to cause harm to other accounts or disks.

In order to protect the machine from unauthorized fortran code inserted by a user into the PELLPACK .e file, specialized filters have been built into the original PELLPACK system. The original PELLPACK language processor already restricted the location of Fortran code to specialized segments within the PELLPACK problem definition file; these segments are now re-parsed by filters that identify inserted Fortran statements for unauthorized code.

Every user is provided with a unique directory for uploading and downloading files, thus facilitating the option of saving and retrieving material. This directory is created by the CGI script after the registration information is entered and validated. User's directories are password protected, securing each user from all other users. Every user file, however, is opened and checked by WebPDELab for legal content as it is uploaded or saved by the user from inside PELLPACK.

2 WebPDELab Features and Issues

In this section, we list the significant benefits resulting from the implementation of the WebPDELab server described in Section 1:

- *Generality.* Any machine connected to the Internet can use the PELLPACK environment without concerns about language or machine compatibility.

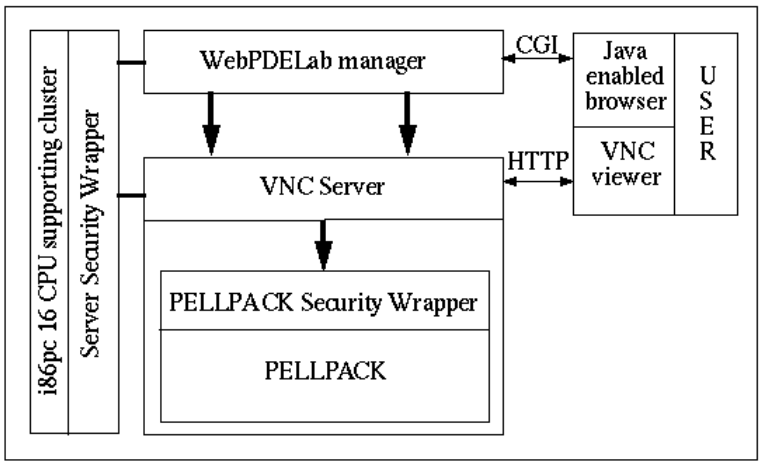

Figure 9: Implementation of the WebPDELab server.

- *Interaction.* Users can specify the PDE with normal interaction speeds for the client machine, since data entry is done locally. The amount of code exported to support the user interface is substantial (several megabytes), but it is only a fraction of the PELLPACK system. If the user has no graphics capability, then the text based interface tools must be used; these are less convenient but still practical to use.

  As the PDE problem is being specified, information is sent to the server. The server might request additional information but once the problem is completely specified, it is solved on the server's host machines. After the PDE is solved, the user can either view output generated by the server or request that the solution (normally a large data set) be sent for local use.

- *Access to High Performance Computers.* Any user can access machines with sufficient power to solve the PDE problem. Even if the solution is too large to be sent to the user (or if there are no local visualization tools), the solution can be explored over the net.

- *No Code Portability Problems.* User do not need to have the code in the local machine language, since the software infrastructure operates only on the server's host machines.

There are several concerns and technical issues involved in the service provided by WebPDELab which we now discuss:

- *Performance of the user interface.* There is a clear trade off in user interface performance between exporting code to the user's machine and executing code on the server. Our existing prototype shows that communicating each mouse click back to the server for processing provides unsatisfactory interactive performance due to network delays. Our analysis

indicates that almost all of the interaction can be run locally by exporting a moderate amount of code. The user interface does use tools that are both time consuming to execute and which are too large to export. Examples are MAXIMA (used to transform mathematical equations) and domain processors (used to create meshes or grids in geometric domains). These tools usually require pauses in response even without a network and the added delay due to networks is unlikely to be significant.

- *Security for the server.* While we control the material received from a user, the server is clearly subject to attack. We place the server on a separate subnet and access licensed software through a gateway. Since we know exactly what is to be sent via an RPC, it is possible to protect this licensed software. Even if a user succeeds in becoming "root", access to other machines is not possible. Of course, network file systems and similar tools are not used. Our process of "registering" users when we give them accounts provides us with a chance to screen users before providing them access to WebPDELab.

- *Security for the user.* This requires each user to be completely isolated from all others. Each user on the server runs in a virtual file system using a login with no access privileges. Thus, each user appears to have the entire machine, and the protection mechanisms between machines protects users from one another. This approach provides security at the cost of using much more memory than normally necessary.

- *Software ownership and fair use.* We prevent the copying of software by placing, if necessary, source code on another machine or another network and using secure RPC.

- *Payment for computing services.* The WebPDELab server is provided free to users as well as time on associated servers used for security purposes. We do not foresee a need to charge users for time on these machines. If large numbers of users contend for service then they will be queued and the cost of the servers is clearly limited. However, there is a real problem when we access parallel machines which act as compute servers. Initially, WebPDELab uses local machines (a 140 processor Paragon, a 64 processor SP-2, a PC cluster with 32 PCs, an SGI Origin 2000 with 32 processors) and a user can easily pose a problem that uses several hours on one of these machines. We intend to access off site machines in the future. When the usage of these compute servers becomes a problem, we will require users to obtain accounts on them. This is a nuisance now but we believe the Net infrastructure will evolve soon to simplify such administrative problems.

There are three technical issues considered in the deployment of WebPDELab as a successful server. First, the user interface must be clearly separated from the rest of the system. Our system is very modular in nature and we have already essentially completed this task. Second, we must create an efficient, exportable user interface. We have already made a prototype exportable

user interface which is neither efficient nor general. It assumes the user has an X-windows server and it requires excessive network communication. We have studied Java implementation, and believe we can use it to obtain both efficiency and generality on the network.

Third, it is the problem of dealing with the visualization of very large data sets over the network. Using WebPDELab, a person with a simple PC can generate a PDE solution consisting of millions of data points in 3-D. In our own group we have 155 Mbit/sec ATM networks and expensive graphics workstations to visualize such solutions. We see two ways to provide visualization service to the user neither of which is always satisfactory. (1) We have visualization tools to slice, rotate, color, etc., data for viewing. We can send these images back over the Net. But the user might have a slow network connection or a black and white display, and in this case the viewing process would be painfully slow. (2) We can send the data set to the user. A two million point solution is not rare and its data set would be at least 25–50 Mbytes. The transmission time could be prohibitive if the user has slow network connections. In addition, the user might not have space to store the solution, or might not have any visualization tools that can handle the data. We believe that visualization over the Net will be a serious problem for some users, and it is one we currently have no solution for. We believe that this is a common problem and that the Net infrastructure will provide solutions in a few years.

3 Summary

In summary, we have an operational prototype of WebPDELab and a plan providing a very useful and innovative network service using it. The implementation of the plan does not require new science or technology and it can be accomplished with reasonable cost and time.

References

[Catlin et al., 2000] Catlin, A. C., Weerawarana, S., Houstis, E., and Gaitatzes, M. (2000). The PELLPACK User Guide . *Technical Report, Dept. of Computer Sciences, Purdue University*, page to appear.

[Houstis et al., 1998] Houstis, E., Rice, J., Weerawarana, S., Catlin, A., Gaitatzes, M., Wang, K., and Papachiou, P. (1998). PELLPACK: A Problem Solving Environment for PDE-based Applications on Multicomputer Platforms. *ACM Trans. on Math. Soft.*, 24, No. 1:30–73.

[Richardson et al., 1998] Richardson, T., Stafford-Fraser, Q., Wood, K. R., and Hopper, A. (1998). Virtual Network Computing . *IEEE Internet Computing*, 2, No. 1:33–38.