Development of a Web Based Simulator Engine for Programmable Software Environments

V.Kannan*, M. Damodaran** and C. W. See+
Singapore MIT Alliance (SMA)
c/o National University of Singapore
7th Floor, Block E3A, University Hall, 10 Kent Ridge Crescent, Singapore 119260

Abstract: - The rapid developments in the internet today have led to our perpetual dependence on the worldwide web for information, processing and computation. The need for quality computational softwares for intensive mathematical computations and simulations motivated the development of packages like MATLAB, MATHEMATICA etc. that allowed users to run complicated routines in almost no time. With the development of internet technology and high tech web methodologies, it was only a matter of time for these powerful tools to be available for operation from a remote location, where the user can run programs residing on a server thousands of miles away and dissect the results. This project is a small step in that direction and aims to develop a robust web based simulation engine through the use of these packages as well as some others like the still nascent Java Numeric Library technology.

Key-Words: - Web Simulator, MATLAB, MATHEMATICA, JNL, Servlet, JAVA, Client-server

1 Introduction
The Web Based Simulation Engine encompasses the culmination of a long cherished human dream – to be able to control the operation of a remote machine and use the processing capacities of this remote machine from anywhere in the world. The project aims to develop a web interface that will allow a user connected to the Internet to access and run a program/simulation from a remote server and will then display the results to the user on the web page.

In doing so, the project involves the use of the now-famous Client-Server Architecture wherein the user machine is the client and sends requests regarding the program that has to be executed, to the remote machine or sever that is located elsewhere.

The server receives and parses this request and then executes the required files to send the final answers/results back to the client machine. The paths or pipelines that have been used for accomplishment of the same are Java and CGI.

Similar efforts in the past involved the use of tools such as ClibPDF [1]. Use of RPC (Remote Procedure Calling) for remote control has also been investigated [2]. The method that we propose to use is outlined in Fig. 1.

2 Technology and Implementation

There are 3 main contemporary software packages that have been studied in detail, namely...
MATLAB, MATHEMATICA and JNL. The implementation methodology differs for each of the the Common Gateway Interface (CGI) in association with the MATLAB Web server feature. The JLink technology for information and data flow through totally Java based component and its utility is exploited in association with Java Servlets. license agreements are met and respected by the user the APACHE server that is used for the development and deployment of all of the above

2.1 MATLAB 6.0 ships with the powerful WebServer feature that enables the client to remotely run a simulation at the server side [3]. programmed to receive requests from the clients through a web interface (developed using HTML). s sent to the server side through the HTTP CGI pipeline and the necessary commands or scripts are run in the server The program is executed on the server and the results are returned once again through CG -HTTP

The server uses CGI to communicate and programs in the server side, in this case, MATLAB. MATLAB executes the required execution sequence interface again through CGI.

The MATLAB Web server encapsulates the CGI and MATLAB part of the above figure, Fig. 2 into a single entity so that the user can communicate with MATLAB and run various scripts. An interactive window for user needs can also be created.

2.2 Java Servlet Technology
The MATHEMATICA and JNL engines are built totally on Java technology. The engines use the servlet technology purely or in combination with some other package, as discussed in the following sections.

2.2.1 Introduction to servlets
Servlets are arguably the most widely used Java applications. The rise of server-side Java applications is one of the latest and most exciting trends in Java programming. The Java language was originally intended for use in small, embedded devices. It was first hyped as a language for developing elaborate client-side web content in the form of applets. Until recently, Java's potential as a server-side development platform had been sadly overlooked. Now, Java is coming into its own as a language ideally suited for server-side development.

To put it simply, servlets are just programs that run on the server side and generate dynamic content. The life cycle of a servlet can be summarized as shown in Fig. 3 [4].

![Fig. 3 Servlet Life cycle](image)

2.2.2 Application in MATHEMATICA engine
The MATHEMATICA engine uses the servlet methodology with an inbuilt MATHEMATICA feature, called Jlink, which
enables the inclusion of Java based routines into MATHEMATICA programs [5]. The user input is got using a HTML form. The information is then parsed to the server through the POST method of the HTML form.

JLink is then invoked through the servlet and the necessary code snippet is executed in MATHEMATICA on the server side. The result is evaluated as either a text or image that is then interpreted by the servlet to a form that can be displayed on the client’s browser.

### 2.2.2 Application in JNL simulator

JNL (Java Numeric Library) is an attempt that is still underway to integrate Java’s portability and other powerful features with computation and simulation for solving numeric systems. The engine implements a simple JNL demo to illustrate how JNL can be used in coordination with the APACHE web server to get numerical results for desired problems.

The JNL demo uses the DoubleVector and DoubleMatrix classes of the JNL and develops a solver based on this. The application can also be extended to other pertinent aspects of the JNL such as Complex numbers.

### 3 Results

The simulator page developed is by no means a complete and comprehensive one. It is just an eye-opener to another of the numerous multifaceted applications that the world wide web is capable of accomplishing. Some of the demos that have been shown are simple but illustrative and can be easily extended for more complicated applications as desired. The figure below shows a screenshot of the main page.

![Main page for the Simulator Engine](image)

**Fig. 4** Main page for the Simulator Engine

The subsequent sections outline and display some of the simulator engines: MATLAB, MATHEMATICA and JNL.

### MATLAB

The web page outlines some of the applications that the web engine can be used for. The illustrations to reiterate the methodology of usage of programs remotely, browser directly, while images are converted to gifs by using MATLAB functions that are run on the these gifs are stored at a location where CGI can access them and then serve them to the

The figures that are shown here are some of the application displayed here is one of a PDE solver boundary problem by using a Generalised Conjugate Residual Method.

![Web Based MATLAB PDE Solver](image)

**Fig. 5** Web Based MATLAB PDE Solver
3.2 MATHEMATICA

The MATHEMATICA simulator is based on a congenial relationship between Java servlets and JLink, which acts as the bridge between Java and MATHEMATICA residing on the server. The results of any operation can be evaluated as text or image, as in the MATLAB case. The text answers can be readily displayed to the client. The image, however, has to be processed. MATHEMATICA evaluates the image to a byte array that is then converted to a Java image object using the Java API. The image object is converted to a displayable format using GIF encoders [6]. These encoders provide a sort of streaming to display the image as required.

The application shown here is a simple plotting application, where the user can plot a 3-D function that he/she specifies in the web page. The output is a plot that is displayed on the browser.

- D plotter

Java Numeric Library

The JNL demo uses Java and JNL APIs to size and then solve them using JNL methods.

![Fig. 7 Web Based JNL Solver](image)

MPI

engine can well be extended to a number of areas such as parallel processing and Message Passing from the web, that is, a problem is selected and the returns the results through the client browser. The implementation of the MPI code can be done MPIJava executable file that implements the code in MPI mpirun or any similar command line server on the UNIX machine or on a suitable machine that communicates with the UNIX
The implementation methodology for MPI is similar to the MATHEMATICA case in that a web server is used to build and implement a page whereby the user can choose the desired MPI program to execute. The program is then launched by the Java servlet. The servlet calls the appropriate program using \texttt{mpirun} and the program then executes on the UNIX machine. The results are then displayed to the user using the servlet again. The information flow is depicted in the following figure.

Fig. 8 MPI Implementation using Servlets

The problem that has been tested is the application of MPI to a standard seven-point finite difference discretisation of the 3D Poisson Equation in a three dimensional enclosure formed by cutting a square hole in a unit cube. The boundary conditions are specified.

The problem is implemented in MPI and then called from a servlet by the user.

4 Conclusion

The Web Simulator is designed to herald the development and necessity of remote simulation through the Internet and is among the various attempts that have been made, and are being made in this respect. The applications and the deployment of technology for this purpose is merely a demonstration of things to come. The application of these technologies to other packages like MAPLE and to totally different technologies like MPI and Parallel Processing cannot be too far away and work is underway in this regard. With the development of these tools and with further work for the above-discussed applications, remote simulation and web-based engines will surely become an inevitable part of the not too distant future.

Acknowledgements:

The authors would like to thank the Singapore MIT Alliance and SGI Singapore for their support and cooperation.

References: