Web Based Framework for Distributed Computing*

Zhikai Chen[a] *Kurt Maly*[a] *Piyush Mehrotra*[b] *Praveen K. Vangala*[a]
Mohammad Zubair[a]
[a]Computer Science Department, Old Dominion University, Norfolk VA 23529 USA
{chen_z, maly, praveen, zubair}@cs.odu.edu

[b]ICASE, MS 403, NASA Langley Research Center, Hampton VA 23681 USA
pm@icase.edu

Abstract

Parallel and distributed computing on a cluster of workstations is being increasingly applied to a variety of large size computational problems. Several software systems have been developed that make distributed computing available to an application programmer. However, these systems either are not web-based or lack a collaborative environment. The increasing use of Web technology for Internet and Intranet applications is making the web an attractive framework for solving distributed applications, in particular, because the interface can be made platform independent. In this paper, we describe JAVADC a Web-Java based framework for the execution of parallel SPMD applications which use PVM, pPVM, and MPI. We also discuss the design of a collaborative, distributed computing environment, Arcade, focussed on more general programming paradigms for multidisciplinary applications. Arcade is a web-based integrated environment which provides support in all phases of the development of general multidisciplinary applications including the design, execution, monitoring and control of such applications.

1 Introduction

Parallel and distributed computing on a cluster of workstations is being increasingly applied to a variety of large size computational problems. Several software systems have been developed that make distributed computing available to an application programmer. We distinguish between two types of run-time environments. Examples of the first category of environments include PVM [1], pPVM [2] and P4 [6]. All these environments support distributed computing in varying degrees of generality; however, they are not web based, they lack collaborative features, and they are mostly suitable for running SPMD programs. The second category of environments address large distributed heterogeneous applications. Examples of such environments include FIDO [3] and MIDAS [4]. However, both these systems are either hardwired to a specific problem or are too restrictive. The other major limitation is that they lack a collaborative environment which allows different members in a group to interact with the application at various stages of its design and execution. There is a Web based computing environment, WebFlow, that enables distributed applications over the Web [5]. The WebFlow, an attractive working environment for parallel distributed applications written in Java, has many of the same goals as our system. However, it does not seem to be focussed on the comprehensive support that we would like to provide through the systems described here.

In this paper, we describe two systems JAVADC and Arcade. JAVADC (JAVA for Distributed Computing) is a Web-Java based framework for executing SPMD applications on workstation clusters. The Java

*This work was supported by the National Aeronautics and Space Administration under NASA Contract No. NAS1-19480 while the authors were in residence at the Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA 23681. The work of the authors from Old Dominion University was also supported by NASA under Contract No. NAG1-1550.

interface is integrated with the Netscape Web browser to keep the user interface familiar and easy to learn. The use of Java makes the graphical interface portable to different platforms. Using JAVADC, a user in one Internet domain can configure a parallel environment on a workstation cluster in another domain, run an application on the cluster and monitor its progress.

The JAVADC environment focuses on SPMD applications and does not provide support for more distributed heterogeneous computing applications. Such applications, for example, the multidisciplinary design optimization of an aircraft, generally consist of multiple heterogeneous modules interacting with each other to solve an overall design problem. Typically these modules are developed in different disciplines and are optimized independently. The traditional way of integrating these modules and optimizing them for the overall design is a long and tedious process (typically taking several weeks). The slowness of this process is mainly due to the absence of a collaborative environment where (i) different modules and their interaction can be specified, and (ii) testing, monitoring, and steering of the overall design can be done.

To address these problems, we are designing and implementing a collaborative distributed computing environment, Arcade. As in the case of JAVADC, Arcade uses web technology to provide a portable and easy to use environment for the development, execution and monitoring of complex scientific applications on a distributed heterogeneous network of workstations and parallel machines.

In the next two sections we discuss JAVADC and the design of Arcade, respectively, while the last section presents some conclusions that we can draw at this stage of the project.

2 JAVADC

JAVADC – JAVA for Distributed Computing – is a web-based environment for managing the execution of parallel and distributed SPMD applications written using MPI, PVM and pPVM [2][1]. Consider a situation where a user in one Internet domain wants to: (i) configure a parallel environment on a high-performance workstation cluster, HPC, in another domain, (ii) run an application on the HPC using executables and input/output files located in another domain, and (iii) monitor its progress. JAVADC has been designed to support this process, independent of the architecture of the user's host machine. The architecture of JAVADC, illustrated in Figure 1, consists of four basic components:

- The *Web Client* is a graphical World Wide Web browser, with a runtime Java environment. For our prototype we are using the Netscape browser. The *JavaClient* is a Java based graphical user interface which runs under the browser environment. The JavaClient interacts with the user and communicates with the JavaServer over the network. It provides facilities for specifying the application and the resource requirements for the target communication systems, including MPI, PVM and pPVM. The client also supports drag and drop feature for running the applications.

- The *Web Server* is a HTTP-server. We are using the public domain Apache server. A *CGI Script* gets executed on the server when a user accesses the JavaClient interface link from the JAVADC home page. The script starts the *JavaServer* if it is not running already. The *JavaServer* is a Java-based server which is the main working engine of JAVADC. It collects the user's specifications and executes the application on the High Performance Cluster.

- The *HPC*, High Performance Cluster, is the cluster of workstations which forms the execution environment for the parallel application.

- The *Application Sites* are the sites where the application executable and the data input/output files are stored. They can be the same as the HPC domain. However, in the JAVADC environment, the application sites maybe at different domains and the JavaServer moves the files, as needed, for execution.

[1] The pPVM runtime system extends PVM to transparently exploit multiple, heterogeneous networks in a workstation cluster.

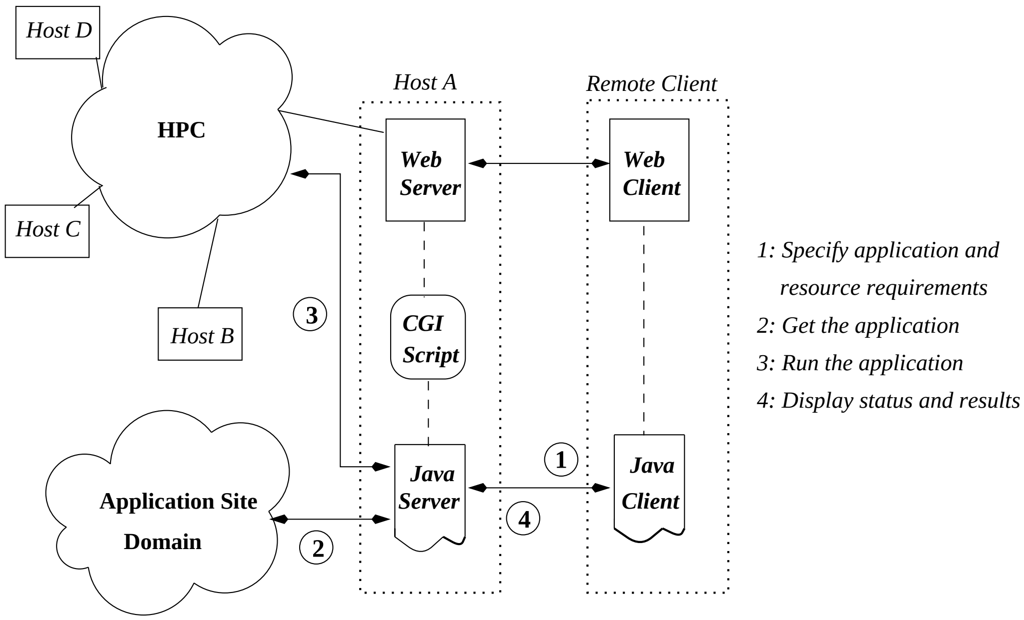

Figure 1: Basic Components of JAVADC.

We have implemented this architecture and successfully run scientific computations using multiple domains. For example, we have launched parallel applications from a PC in the odu.edu domain with all files being located in that domain, run the computation on the HPC in the icase.edu domain and monitored the results on the PC. We did not observe any significant performance degradation due to the Java server, client communication and execution. A serendipitous result was how code developers used the system to analyze configuration impact on performance. They launched several applications on the HPC, identical in all respects except the number of machines used, and monitored the timing results in separate windows.

3 Arcade

The Arcade environment extends the focus of JAVADC to more general applications, such as the multidisciplinary design optimization of an aircraft. Such applications in general consists of multiple heterogeneous modules interacting with each other to solve the overall design problem. Arcade is an integrated environment designed to provide support for a team of discipline experts to collaboratively design, execute and monitor multidisciplinary applications on a distributed heterogeneous network of workstations and parallel machines.

The system consists of the following sub-environments:

- The *application design interface* will allow the hierarchical specification of execution modules and their dependencies. The system will support multiple mechanisms for specifying the application, including textual scripts and visual interfaces.
- The *resource allocation and execution interface* will provide support for specifying the hardware resources required for the execution of the application. The resources could be chosen by the users or by the system based on the current and predicted loads of the system and the characteristics of the application. The choice could be made statically or dynamically during the execution of the application.
- The *monitoring and steering interface* will allow multiple users not only to monitor the execution status of the application but also to examine intermediate and final results including collaborative

visualization of large data sets. The system will also allow the users to steer the execution of the application in various ways, e.g., by changing the values of arguments to modules or by exchanging plug modules to change the fidelity of the simulation.

The overall goal is to design an environment which is easy to use, easily accessible, portable and provides support through all phases of the application development and execution. We plan to leverage off of existing technologies to implement various parts of the environment. This includes not just Java and browser technologies but also GUIs, communication APIs, and other environment independent modules from existing systems. We propose to have an open architecture providing APIs to integrate other technologies within our design. For example, using the APIs it will be possible to integrate different application design specification tools within our environment. Wherever possible we will use this approach to use off-the-shelf technologies to build the complete system.

3.1 Arcade Architecture and Status

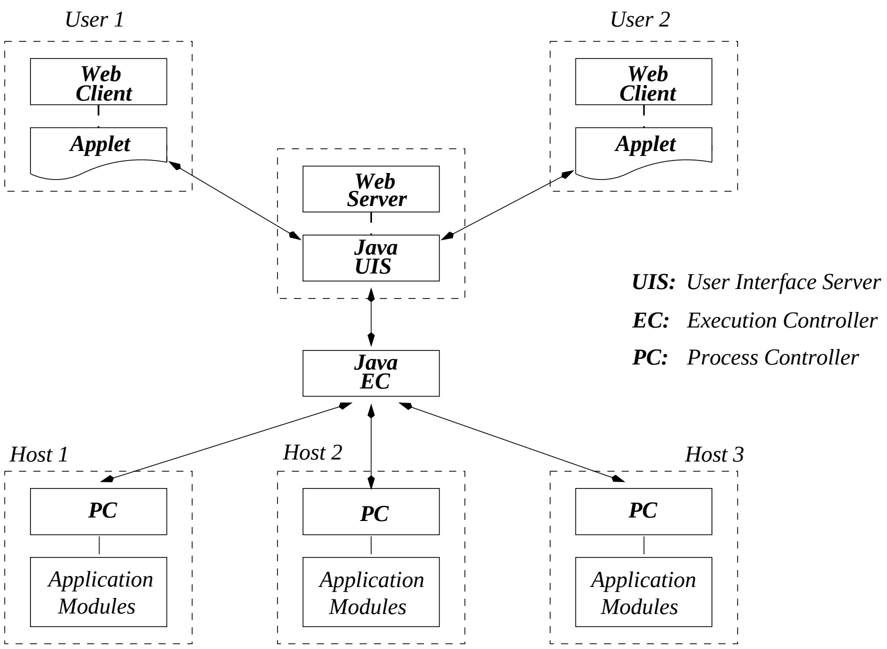

Figure 2: Architecture of the Arcade System

The Arcade system, as shown in Figure 2, is a three-tier architecture. The front-end, first tier, of this system is a GUI interface integrated with the Web. It is implemented using Netscape Internet Foundation Classes (IFCs). Integrating the interface in a web browser, e.g., Netscape, will provide users with a familiar interface on desktops ranging from Unix based workstations, to Windows-based PCs, and MacIntoshes. Typically, this interface is invoked by downloading a URL document. The Java based applet is downloaded along with the document and opens an interface for the user after an authentication process. The interface consists of the sub-interfaces for application specification, resource allocation and execution, and monitoring and control, as described above.

The middle tier consists of logic to process the user input and to interact with application modules running on a heterogeneous set of machines. The overall design is a client-server based architecture in which the *User Interface Server (UIS)* interacts with the front-end client to provide the information and services as needed by the client. When the user requests the execution of an application, the UIS launches an *Execution Controller (EC)* which manages the overall execution of the application by firing up user modules on the specified resources as and when required. The UIS and EC also cooperate to service any monitoring and steering requests from the client.

The EC interacts with lightweight processes called *Process Controllers (PCs)* running on the individual resources in the execution environment. The PCs along with application modules form the last tier of the architecture.

The main advantage of a three-tier system is that the client or the front-end becomes very thin, thus making it feasible to run on low-end machines. Also, since most of the logic is embodied in the middle tier, the PCs can be kept lightweight thus keeping the additional loads on the executing machines to a minimum.

We have implemented the three-tier system as described here in a prototype Arcade system. The current system allows single users to specify non-hierarchical applications through a textual or a simple visual system. The resources have to be statically specified by the user. The system supports the execution of the modules on a network of workstations in a single domain. The execution status of the application can be monitored by multiple users simultaneously. We are currently expanding the system to incorporate all the facilities envisioned in our architecture in a phased approach, an approach where the driving force is the user of the system.

4 Conclusion

In this paper we have discussed JAVADC which enables users in one Internet domain to run a distributed application on a workstation cluster in another domain. We have successfully implemented the architecture and tested the multi-domain separation of source, data, compute engines, and monitoring engines. Although we did not perform a formal evaluation, we do have anecdotal evidence that users prefer our system to the old fashioned combination of telnet, ftp, rpc, and X display. There certainly was no noticeable degradation in performance. Actually, it was the success of this project which led us to the design of Arcade because it convinced us that web technology is the right way to manage and steer large distributed applications. Arcade is an environment to design, execute, monitor, and control a distributed heterogeneous computing application. At this stage we have a rudimentary architecture implemented which supports the execution of multiple applications and their monitoring. Even just these features have created quite an interest in this approach and encouraged us to continue with the implementation.

References

[1] V. Sunderam. "PVM: A Framework for Parallel Distributed Computing," *Concurrency: Practice and Experience*, Vol 2 No 4, December 1990.

[2] K. Maly, S. Kelkar and M. Zubair. "Scientific Computing using pPVM," *International Conference on Parallel Processing*, vol 2, pp. 201 - 205, August 1994.

[3] R. P. Weston, J. C. Townsend, T. M. Eidson and R. L. Gates. "A Distributed Computing Environment for Multidisciplinary Design," *5th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, Panama City, FL, AIAA 94-4372, September 7-9, 1994.

[4] John C. Peterson. "Multidisciplinary Integrated Design Assistant For Spacecraft (MIDAS)," http://mishkin.jpl.nasa.gov/Midas_Page.

[5] Bhatia, D., Burzevski, V., Camuseva, M., Fox, G., Furmanski, W., and Premchandran, G. "WebFlow–a visual programming paradigm for Web/Java based coarse grain distributed computing," *Concurrency: Practice and Experience* , March 1997. Java Special Issue. Guest Editor: Geoffrey Fox. (Also available as Syracuse University Technical Report SCCS-78).

[6] R. Butler and E. Lusk. "User's Guide to the P4 Programming System," Technical Report ANL-92/17, Argonne National Laboratory, 1992.