Chapter 6

Concluding Remarks and Directions for Future Research

6.1 Summary and Implications of Thesis

Recent progress of HPCC technologies is toward building high-performance parallel and distributed computers from commodity workstations and networks. Supported by this trend, the last couple of years have seen the concept of metacomputing reach widespread acceptance as an effective alternative to other, more traditional computing paradigms. This is mainly due to the rapidly falling price-performance ratios of networked computing resources and the emergence of high-speed local- and wide-area networks. Such a metacomputing environment consists of a hardware mixture ranging from clusters of workstations to parallel supercomputers, accessible as a single operational unit via geographically distributed networks.

Although the World-Wide Web was initially designed to be used as an on-line multimedia document retrieval system, it comes with a much more powerful infrastructure that could be used for other purposes. The following four points about the Web infrastructure could contribute to building metacomputing environments:

1. WWW is a network-based, distributed environment by definition.
2. Web (i.e., HTTP) servers provide a standard open interface to access every machine in a similar way, regardless of its actual hardware architecture or software configuration.
3. Web servers can be configured to have full access to all the high-performance computational resources on the same domain. With the addition of other server-side Web technologies, such as CGI and LiveScript, they can be extended as computation and coordination servers instead of document servers.
4. Graphical Web browsers can be thought of as portable, open, operating-system-independent, graphical user interfaces. The new Web technologies offer fresh possibilities for the high-performance computing community to adapt these interfaces to all the different computing platforms (including PCs, Macintoshes, and Unix machines).

Given these factors, incorporating emerging Web technologies into high-performance computing and communication technologies looks promising. Such an integrated system would combine the strengths of both technologies and would help to transfer the last decade's high-performance experience onto a new platform. The focal point of this thesis is building a Web-based metacomputing environment combining both technologies.

The World-Wide Virtual Machine (WWVM) combines geographically distributed high-performance computing resources on local-area or wide-area networks or on the Internet and provides a metacomputing environment. It provides users with the illusion of a single, large machine that is remotely accessible over the Web using the Web browser interface. Delivering computational applications over the Web in an interactive manner using off-the-shelf Web browsers as the interaction medium is intuitive and has the potential to attract a large user community, improve the rate of user acceptance, and avoid many of the pitfalls of software distribution.

WWVM requires at least one active WWVM server to present in each site or computational domain in order to coordinate the WWVM operations at that site.[1] A WWVM server is a CGI-extended form of a conventional Web server. Local WWVM servers running on these platforms can access all the computational and information resources on the server machine and all other machines connected to this server machine within an organization. Such a server's duties include, but are not limited to, acting as a computational server when required, helping to configure and manage the local machines in that site, participating in code and data transfers between the local site and other sites, and providing a uniform standard interface to the users.

Any individual machine that is part of the WWVM configuration can be used in stand-alone mode. This allows choosing, for example, a specific workstation cluster to attack a particular type of problem. In addition, a subset of the available hardware resources can be used at the same time to build a virtual computing platform that can operate in two different modes: message-passing mode for parallel processing and dataflow mode for distributed processing. The important points of these three modes of operation are summarized below.

- *Stand-alone mode.* Web-based parallel programming environments (WPPEs), described in Chapter 3, are small-scale applications of the stand-alone mode of operation to the education domain. WPPEs are unusual among Web services, because they allow users to create, edit, and execute files, rather than simply retrieve them by following hypertext links or by making simple database queries. WPPE prototypes that provide support for high-level parallel programming based on Fortran 90 and High Performance Fortran (HPF), as well as explicit low-level programming with the MPI message-passing interface and traditional serial programming, were field-tested at the Cornell Theory Center and at Syracuse University. These prototypes contributed to our understanding of the critical issues related to user interface design, preferred environment settings, and the facilities necessary in such an environment.

- *Message-passing mode.* This mode allows the tight coupling of selected machines on local- and wide-area networks as a single parallel-computing platform.[2] It should be noted that the bandwidth of the Internet that can currently be achieved inhibits the effective use of WWVM as a parallel computing resource for communication-intensive applications. For this reason, it is assumed that high-speed WANs are available for interconnection when two machines at distant locations are to be connected. The provision of this mode required being built on the software environments and with the distributed computing capabilities provided by the Web, merely adding the low-level communication and coordination needs of parallel processing by using modified PVM daemons. The computers on a network are usually from different vendors, and through the use of PVM the WWVM system can cope with differences in the architecture, data format, computational speed, machine load, and network load.

  The WWVM message-passing mode provides multi-library and multi-language support. Since the underlying communication layer is based on PVM, a high-level interface to MPI is provided that emulates MPI functions in terms of PVM calls. Furthermore, message-passing

[1] Since Web servers are locally managed, local sites may easily change management policies concerning the use of the site's resources. All hardware resources in a site should be accessible through a common file system such as NFS or AFS.
[2] This is similar to the IWAY effort, where MPI is extended to support inter-site communication.

wrappers from Express and TCGMSG message-libraries to MPI enable the running of programs based on those two libraries on top of WWVM. HPF and Global Arrays shared-memory programming models are also supported by the WWVM. The Syracuse Fortran 90D/HPF compiler and the PCRC runtime support libraries are used to compile and run HPF programs.

- *Dataflow mode.* In the dataflow mode the independent tasks of a large problem are distributed to several machines on different sites. According to the problem's task dependency diagram, the output of a task is fed into another task's input by using HTTP-based data channels. Since each task is independent in its operation, it is assigned to the type of machine that will support its requirements (different programming paradigms, architecture types, etc.). Therefore, an interface layer similar to the one found in the message-passing mode is not required.

  The dataflow model supports coarse-grained software integration, and is therefore insensitive to deficiencies in the current Internet or to high overheads of Web software. Even under the high communication cost and network delays, the dataflow model works fine for a subset of high-performance computing applications.

WWVM also takes care of other metacomputing-related issues such as management of heterogeneity, reliability and fault tolerance, dynamic load balancing, security, scalability and expandability, communication across networks, functionality, and providing a shared file space by moving data and core to the required site. These concepts were discussed in various sections of Chapters 2 and 3.

The amount of trace data produced in parallel systems with multiple processors can be overwhelming and difficult to interpret. Visualization is a proven method for dealing with large volumes of complex data. Support for data and performance visualization is an important component of a metacomputing environment. WWVM performance and data visualization components were completely written in a platform-independent manner using Java, which ensures that accessing the same displays can be done easily by using personal computers or high-end workstations. Below is a summary of the main features of the performance and data visualization components of the WWVM:

- *Performance Monitoring and Visualization.* The first step in the performance visualization process is to instrument the program and collect the performance data that will provide insight to the behavior of parallel programs (i.e., the internal course of events taking place) in order to improve performance. This software instrumentation can be in the form of macros inserted at compile time or embedded in system libraries (e.g., interprocessor communication) and invoked at run time. The WWVM system uses the execution trace files generated by the Pablo environment for performance analysis and visualization. Pablo generates a self-describing data format (SDDF) trace file that includes a header describing the structure and semantics of the data in it. The WWVM system uses a simple parser to interpret the header and determine the execution trace record types. Analysis of the records provides information about the impact of CPU utilization, network utilization, load balancing over the whole system, and overhead induced by communication between the interacting processes that are the main attributes of a distributed program behavior. WWVM provides multiple windows that permit several simultaneous views of one parallel program, which allows an analysis of the same program from a different perspective. These windows can be grouped into four main categories: utilization, communication, task, and input/output, and they include textual representations, time process diagrams (Gantt charts), and animations of program execution.

- *Data Visualization and Wrapper Functions*. WWVM has a separate component for plotting data stored in a file as two-dimensional graphics such as line graphs, scatter plots, bar charts, and contour plots. In addition, application-specific displays of data can be constructed at real-time using data wrapper functions. Data wrappers provide a general interface between running message-passing and HPF applications on the server-side, and visualization and control applets on the client-side. They help to interactively display data and control the flow of the program. Two primary models of operation are supported. In the *push model*, the control of the initiation of data transfers belongs to the parallel program. The parallel programs push data to an external client-side Java applet that always waits for the data to arrive over a communication channel. In the *pull model* an external client operating on the server-site accepts requests issued by the Java applet and serves them.

6.2 Directions for Future Research

There are two key directions for future extension of the research presented in this thesis. First, the WWVM framework can be extended to include Java-based computation in addition to the message-passing and dataflow computation paradigms using traditional libraries and languages. Second, the nature and scope of the WWVM architecture could potentially be expanded toward a more layered, modular, and portable architecture and toward more comprehensive support tools.

The WWVM system represents a client-server computing model that employs at least one CGI-extended Web server on each remote site that is responsible for coordinating the computational activities in that domain and for providing a relationship with the rest of the WWVM system. Local computations are in the form of programs written using traditional languages such as C, Fortran, and HPF, and may use message-passing or dataflow computing paradigms. However, the introduction of new client-side Web technologies such as Java technology has enabled the production of a *client-client* computing model. With little effort, the WWVM can provide an infrastructure that will perform the same tasks carried out by Java-based metacomputing systems such as SuperWeb, Charlotte, and DAMPP. Platform-independent and secure applets that report their completion status to a central WWVM server may be activated through the Web browser interface. However, as discussed in Chapter 2, applicability of those systems is limited to a very small number of specific massively-parallel scientific applications with little or no inter-process (or inter-task) communication. Currently, the lack of Java compilers, the slow speed of current Java interpreters, and the long latency of the Internet negatively affect the loading of large Java applet class files by remote clients. Furthermore, the task-launching mechanism in those systems requires a rendezvous between the server and clients. The Java applets are transferred to and run on the client's machine only when a client connects, using his Web browser, to the server. In contrast, a central WWVM server can launch jobs on volunteer sites that have already registered by sending job requests to the WWVM server on their domain. Simultaneous start-up of as many tasks as required makes it possible to get timely results. Since users' identities are verified by using an authentication mechanism, Java applications that are capable of doing input/output and inter-process communication could be used instead of Java applets.

There is also a trend toward writing translators to convert legacy programs written in traditional languages to Java. In the WWVM environment such a translator may enable the distribution of tasks to Web clients that connect to the WWVM through a Web browser.

Other key capabilities and supporting tools of the WWVM that are subject to more enhancements and extensions are discussed below in the rest of this section. As Web technologies continue to evolve, the WWVM design will naturally take advantage of those new developments as appropriate. An investigation is certain to be made of the most recent developments, such as ActiveX, which can carry much of the work from the server-side to the client-side.

Use of Java-based Web servers looks more promising than their traditional counterparts because of their extendibility, interoperability with other distributed computing technologies, and platform-independent implementations. Currently, Java-based servers are still in the development and testing stage and are not in widespread use. Only the Web servers that are well tested and verified[3] for security are used in the WWVM environment, since server software security is very crucial in order to keep the integrity of the systems on which they are installed. A security flaw may cause external hackers to take control of the entire WWVM system. On the other hand, experimental Java servers still lack such a detailed security verification, which makes them more vulnerable to security attacks.

Also intended for the future is the creation of a more flexible WWVM system that is easily manipulated to meet the local system and software requirements. Extensions could include new facilities for more intelligent scheduling of user jobs and for giving users more control of the manipulation of submitted jobs.

Tools for choosing lightly-loaded computational servers for running users' programs automatically, submitting batch jobs from the same interface, and detecting and terminating run-away user processes can be implemented. For these purposes, the use of EASY-LL looks appropriate, at least on IBM platforms. EASY-LL is a scheduling mechanism for LoadLeveler[4] jobs that was developed jointly by the Cornell Theory Center and IBM and is available to other

[3] A few bugs hindering the security were previously discovered and eliminated in current implementations of NCSA, CERN, and Apache servers used by WWVM.
[4] LoadLeveler is a batch system IBM SP-2 and RISC System/6000 clusters that allows the system to be divided into interactive and batch sites, and controls the scheduling of parallel and serial jobs on nodes.

sites for installation. It provides fair scheduling for both large and small jobs by means of a "backfill" algorithm that allows small jobs to make use of idle nodes.

It is clear from users' feedback that the ability to use an editor of their choice is a key component with respect to usability. An important intention is to provide an integrated development environment that alleviates the necessity for the user to learn a great deal of detailed information in order to be successful. It is essential to provide a customized visual Java editor in addition to the currently supplied CGI-based and conventional UNIX editors. This Java editor could be used from home PCs as well as from UNIX workstations, while many UNIX editors can only be called in X-Windows environments. Extended editor functions for the insertion of appropriate data wrapper functions or performance instrumentation calls into users' programs are also planned.

WWVM's visual file manager component requires little further development. With the arrival of wider support for Java 1.1 by mainstream browsers, its drag-and-drop capability and keyboard action shortcuts could be exploited to provide a more user-friendly tool for file operations. The WWVM system currently lacks a visual interface for describing tasks and specifying data dependencies between tasks, now defined using text descriptions. It would be more appealing to develop a visual interface such as the one in WebFlow [FoxF 96b] that shows the system state and task relationships.

Java should grow in importance as a pervasive client-side technology for building interactive analysis and visualization capabilities. The present performance visualization capability is built around the Pablo technology developed at the University of Illinois at Urbana-Champaign. This utility comprises a set of Java applets to support the Pablo SDDF format. This capability could be extended to include other performance analysis tools and to provide an interface that allows users to choose their favorite tool.

Although basic data visualization capabilities have been included in the WWVM, more advanced visualization features are still immature. A future target would be to enhance them in order to support more general classes of scientific data representation that, for the most part, require three-dimensional visualization capabilities. It would also be necessary to develop a standard Java applet library to handle the various types of structures that are expected to be encountered in modern scientific applications. This library could be used in constructing the application-specific visualization applets and to help in the interactive steering of arbitrary applications through the Web interface.

The initial prototype implementation for visualizing replicated and distributed HPF data structures provides the basis for future work. In message-passing parallel codes there is no explicit mechanism to specify array structures and their decomposition to which a tool can refer; the user is the only source of information about the decomposed array. An interface that allows the programmer to specify the structure and decomposition of arrays in a program will be provided. Although functions use HPF's own intrinsic routines to access distributed data items, portable runtime primitives need to be developed in order to collect distributed data items from all processors in message-passing codes.

It would be interesting to investigate the possibility of applying the visualization concept to providing an interface to other visualization systems in order to complement an existing framework. Existing general and widely used visualization systems such as IBM Data Explorer, AVS, and SGI Iris Explorer may be available in the target environment and should not be excluded from tool integration. Those tools covering the range of visualization capabilities needed by computational scientists and engineers can be interfaced with the WWVM. The user would still be able to handle the controls from the browser, while the underlying visualization application will run on a server with a more powerful rendering engine.