# Chapter 1

# Introduction

During the last decade there has been an exponential growth in networked computing resources, accompanied by an astonishing increase of computational and storage power for these attached computers. Cheong [Cheo 92] summarizes trends in five key technology areas that drive the high-performance scientific computing:

- The performance of CMOS-based microprocessors has quadrupled every three years since 1985.
- Local area networks have improved by a factor of ten every decade.
- Computer backplane busses have improved by a factor of ten every decade.
- Semiconductor memory chips have quadrupled in capacity every three years since 1972.
- Magnetic disk storage has evolved from a density of 1K bits per square inch in 1957 to 1G bits per square inch in 1990, doubling every three years.

Recent advances in high-performance interconnections and switches decreased communication latencies between two connected systems from hundreds to tens of microseconds. This is close to the traditional massively parallel processors' (MPP) latencies. As high-speed networks become more common, the power of these combined resources may exceed the power of a single supercomputer. In addition, while the large MPPs typically cost a few million dollars, building a distributed computing system from locally existing resources is quite economical.

*Network-based distributed computing,* which is a way of connecting a set of computers on a network to solve a single large problem, has become an alternative to other forms of high-performance scientific computing. Modern computing environments often consist of a hardware mixture ranging from clusters of workstations to parallel supercomputers. *Metacomputing* is a special subset of network-based distributed computing that involves accessing those machines, some of which are accessible via geographically distributed networks as a single operational unit.

## 1.1 High Performance Computing and Communications (HPCC) Initiative

An impressive development in every aspect of parallel and distributed computing technology has been witnessed in the last decade. This is partly because of the Federal High Performance Computing and Communications (HPCC) Program that has conducted long-term research and development in advanced computing, communications, and information technologies, and in applying those technologies to the missions of the participating Federal departments and agencies [HPCC 96]. This federally funded program has focused on developing technologies that can be applied to computation-intensive applications (the *Grand Challenges* of science and engineering) and information-intensive applications (the *National Challenges*) and has become the largest supporter of research and development in this direction during the last decade. As a result, the technologies developed under the framework of this program are known as *HPCC technologies*.

Representative HPCC technologies include high-performance computing systems with large computational power and expanded memory for solving large-scale, complex problems accurately, large-scale networking techniques[1] that have advanced the availability and effectiveness of distributed applications, and advanced software technologies such as parallel languages and compilers, high-performance software tools, and engineering and scientific libraries.

## 1.2 World-Wide Web Technology

If we consider the whole computing market, the user base of different types of computing can be illustrated as a pyramid with narrow top and much wider base. The HPCC technologies have been developed at the top of the computing pyramid, mostly by federally funded organizations, where it was encouraged to move down into the broader user base [Fox 96]. However, the result has never been very satisfactory due to the lack of common open interfaces among personal computers with Windows-based user interfaces and Unix workstations with X-Windows-based user interfaces and utilities. The advent of the World-Wide Web (WWW, or Web) has brought an important infrastructure that can be used by those same high-end users in order to make these HPCC technologies available to the lower levels of the computing pyramid. WWW includes a body of software and a set of protocols and conventions that allow Internet users to access data through the use of a graphical user interface. WWW started an open environment where people can access public information fast and conveniently through Web-

[1] They have also been instrumental in the evolution of the Internet.

based interfaces. The WWW also provides, in a similar way, an operating-system-independent interface to access every kind of machine, whether a Unix workstation, PC, or Macintosh.

The initial WWW prototype has grown rapidly since it was developed at CERN (European Laboratory for Particle Physics) in 1990, and has become the most popular system on the Internet [BCGP 92]. According to an Internet Domain Survey conducted in January 1996, about 76,000 systems have a registered domain name, `www`, up from only 600 in July 1994 [NetW 96].

## 1.3 Problem Statement

The surge in popularity of the WWW has corresponded to a decreasing market for special high-performance computers [YLFG 96]. As indicated by Gordon Bell [Bell 95], the evolution of computing is towards a universal parallel or distributed system of commodity networks and of commodity workstations and compute nodes. Web is a distributed computing resource by definition, and this may be the next platform for the porting of HPCC technologies once developed for high-performance computers.

Fox and Furmanski [FoxF 95, FoxF 96a] describe a way in which the current HPCC technology base can interact with evolving Web technologies. They suggest that if the worldwide system of several hundred million personal computers could be combined then HPCC expertise in parallel algorithms and methodology could be applied to this remarkable parallel computer. They also indicate that it is natural to build virtual global computing environments in terms of pervasive WWW technologies.

The WWW provides a standard open interface for the retrieval of data and information located on a Web server. It is natural to exploit the same interface in order to access the computational resources of the server. Moreover, an HTTP (Web) server can easily be extended with CGI (Common Gateway Interface) modules and converted to a gateway that provides controlled access to other machines' computational resources within the same domain. If the activities of those Web servers could be coordinated in some way, then a virtual global computing or a metacomputing environment combining the computational power of all those machines would become possible.

Although the effective operation of such a computing environment may require more powerful Web servers, this should not be a problem. Since the Web technology was built for the largest market, and investments and resources have flowed into its development in recent years, it is expected that very high-performance Web server software and hardware will become available soon. Those servers would be able to serve millions of requests effectively and even perform some computation-intensive operations efficiently.

The main purpose of this thesis is to build a metacomputer by elegantly using the emerging World-Wide Web infrastructure and exploiting the HPCC technologies thus far developed in parallel and distributed systems domains. The research work explores potential ways to help carry most of the high-performance computing experience of the last decade to the new Web platform, and in which emerging Web technologies can be used to further advances made in the HPCC area. Not only do Web technologies offer fresh capabilities, but their pace of development is unlikely to be matched by the traditional high-performance research community. The crucial issues that may arise while trying to combine the capabilities of the Web infrastructure and HPCC technologies are still not well known or researched. This work is one of the first efforts to address the issues of incorporating Web and HPCC technologies into the same framework for the purpose of building a metacomputer.

## 1.4 Contributions of This Research

The biggest contribution of this research is the building of a metacomputing environment called the World-Wide Virtual Machine (WWVM) that incorporates emerging Web technologies into the technologies developed within the HPCC framework. Geographically distributed high-performance Web compute servers, and other networked machines (i.e., parallel computers, workstation clusters) coordinated by them, are tightly or loosely coupled using UDP/IP and TCP/IP connections, as well as HTTP channels.

In the loosely-coupled mode, which is also known as the dataflow computing mode, each site is independent in its operation, but inputs and outputs of the sites are connected to each other in order to perform a computation in a coordinated manner. Web servers work like supervisor nodes that closely watch over the local computational activities and do inter-node data transfers when necessary. Data and control messages flow from one WWVM server to another using HTTP links.

A tightly-coupled mode employs PVM daemons in addition to the CGI-extended HTTP servers, and it uses daemon-to-daemon UDP/IP and TCP/IP links to transfer data and HTTP links to pass control messages. This is an extension of the widely used message-passing programming model.

The WWVM supports a number of important HPCC software technologies, including the data-parallel HPF programming language, Parallel Compiler and Runtime Support Consortium (PCRC) runtime support libraries, MPI, PVM, TCGMSG, and Express portable message-passing libraries, as well as the task- and data-parallel Global Arrays (GA) shared-memory programming model.

The WWVM also provides a well-developed graphical user interface (i.e., Web browser interface) for accessing remote supercomputers or other high-performance computing machines by connecting to a WWVM server. Computational problems can be submitted to any remote or local machine currently registered as a WWVM resource, and their results can be displayed in a graphical way using the same Web browser interface. The WWVM automatically provides system accounts for users on remote machines. By exploiting the WWVM infrastructure, three representative parallel programming environments were implemented, namely, the *NPAC Fortran 90D/HPF compiler on the Web*, *Web/HPF*, and the *VPL (Virtual Programming Laboratory*.) The first environment was demonstrated at Supercomputing `95 when the concept of delivering HPCC applications was a very new and innovative concept. The NPAC/Cornell Theory Center's (CTC) Web/HPF System was developed in collaboration with CTC to be used by those attending the CTC Virtual Training Workshops. The NPAC VPL was used for teaching parallel programming languages in a computational science course at Syracuse University.

Another important contribution of the WWVM is the natural linkage of components that provide the capability of instrumentation and visualization for a given program's data and performance. A data visualization facility was implemented using Java for plotting two-dimensional graphics of output data from executed programs. A special library called *data wrappers* provided a way to pass data between client-side Java applets and server-side C, Fortran, and HPF programs. Data wrappers can be used to visualize a program's (possibly distributed) data structures and to interactively steer the computations on remote machines from a Java applet. WWVM also supports performance visualization. Pablo [RAM+ 92] is employed for instrumenting the performance of programs and for generating a trace file in SDDF [Aydt 96] format. The trace file is then displayed using either static or animated displays implemented in Java.

## 1.5 Related Work

The work presented here has been influenced by a number of other research projects. This section presents some of those related works.

Isis [BirM 90, BirC 91, FinK 92] was originally developed by Cornell University and later commercialized by Isis Distributed Systems, Inc. It is a fault-tolerant system that supports groups of cooperating processes, replicated data, and distributed computation, and it provides a set of tools for developing distributed applications in Ada, C, Common Lisp, C++, and Fortran.

Polylith [Purt 86, PurJ 89, PurJ 91, PRG 88 ] was developed at the University of Maryland. It is used to prepare applications with mixed-language software components and execute them in heterogeneous environments. It includes a Module Interconnection Language (MIL) for defining an application's structure and geometry. MIL presents programs as a simple graph where nodes correspond to modules and arcs represent intermodule bindings. Connections among modules are achieved using TCP/IP or XNS protocols on a network, or via shared memory on a tighty-coupled microprocessor, or via custom channels between processors.

XPVM [GKP 94] provides a graphical console for PVM. It allows users to monitor and change the virtual machine configuration by adding or removing hosts. In contrast to WAMM, it does not include facilities for source code distribution or for compilation or execution on remote hosts. However, it does provide trace data analysis and visualization facilities. It provides a space-time diagram, utilization count, network animation display, network queue, and two textual displays: call trace and task output. Unfortunately, it gives no statistics or cumulative displays and traces of user events.

Schooner [Homer 91, HomS 92] was developed at the University of Arizona. It provides transparent distribution and heterogeneity based on a common stub module paradigm. It uses a sequential execution model in which control passes from module to module to ensure distributed computation and visualization. Intermodule communication is achieved using a remote-procedure call mechanism. Schooner is made up of three services: an external data represantion for data coercion, a procedure specification language and associated stub combilers, and a run-time system for control flow.

The Computing Center Software (CCS) [Ramm 95, RRK94] of the University of Paderborn, Germany, focuses on the resource access and the allocation problem of metacomputing. It uses a general-purpose, resource-description language for describing system components and the underlying model. A second project, called Metacomputer Online (MOL) [RBD+ 96], of the same institution aims to utilize multiple WAN-connected high-performance systems as a computational resource for solving large-scale problems. It is designed as an open, extensible software system comprising a variety of software modules, each of which is specialized to serve a specific task such as resource scheduling, job control, task communication, task migration, or user interface. MOL provides a generic interface for effective interaction between those modules.

The Argonne National Laboratory is developing a basic software infrastructure called Globus [FosK 96, FKT 96] for computations that integrate geographically distributed computational and information resources. The focus of the project is to build the low-level mechanisms that will be required to implement higher-level services. Several components of the Globus project were demonstrated in the context of the I-WAY experiment at Supercomputing '95.

The Wide Area Metacomputing Manager (WAMM) project involves several sites in Italy [BFF+ 95]. It provides a very powerful graphical user interface to users. WAMM groups connected hosts in a tree-structured hierarchy according to their geographical location on the network. It allows the remote compilation and execution of user tasks on selected hosts by exploiting the UNIX remote command execution mechanism. WAMM also supports message-passing programming with PVM.

The Legion project [Grim 92] of the University of Virginia is perhaps the most developed of the metacomputing efforts. It connects workstations and supercomputers on a wide-area network to provide the illusion of a single virtual machine. It provides an easy-to-use, seamless computational environment, includes resource management facilities, allows parallel processing, ensures security for both users and resource owners, exploits heterogeneity for high performance and fault tolerance, and provides a single, persistent namespace. Legion architecture is object-based with objects communicating via typed-messages. The Mentat language [Grim 91, Grim 93, Stra 92] is used to write programs for parallel and distributed systems. Mentat is an object-oriented language based on C++, but it also provides interfaces for other languages such as C, ADA, and Fortran. Mentat can define medium- to coarse-grained control and data parallelism. It supports only messages between objects, but its compiler can automatically generate the code for all communication and synchronization. Mentat uses a virtual machine model that facilitates the porting of applications. For data-parallel objects, Mentat adds data-partitioning semantics via call-backs and system parameters to dynamically partition a data structure for parallel processing with automatic load balancing. Thorough the call-back definition the user can explicitly control the distribution of data.

NetSolve [CasD 96] is a client-server, network-based system that allows users to access remote computational resources distributed across the network. NetSolve currently offers a number of scientific packages such as BLAS [LHK+ 79, DDH+ 88, DDD+ 90], LAPACK [ABB+ 95], ItPack [YKR+ 96], LINPACK [DBM+ 79], and ScaLAPACK [DonW 95]. NetSolve provides the tools to look for resources on a network, choose the most suitable one for the specific purpose, and launch the computational problem on that resource. It provides easy-to-use interfaces to C, Fortran, MATLAB [MathW 92], and Web users. A set of independent NetSolve systems on different locations provides various services. Each host in the NetSolve system runs a NetSolve computational server or resource. Users contact Netsolve agents via Web browsers to get help in choosing a suitable system to perform required tasks by directly connecting to the chosen system. NetSolve uses a load-balancing strategy to improve the use of available computational resources. Based on the size of the problem, complexity of the algorithm used, and workload and raw performance of a machine, the server selects the machines that can be used. Since NetSolve is a loosely connected distributed system, fault tolerance is also taken care of. Failures due to network malfunctions or server failures are detected, in which case the client sends the problem to another available server. Eventually, unreachable hosts are removed from the system server tables. As opposed to Java-based systems in which the program resides on the server and is downloaded to the user's machine, where it operates on the data and generates the result on that machine, the program resides on the server in NetSolve. The user's data is sent to the server, where the programs or numerical libraries operate on it and the result is then sent back to the user's machine.

### 1.5.1 Web-Based Metacomputing

A majority of the Web-based metacomputing efforts are Java-based: they either use Java as the computation language or they use Java communication and threading mechanisms for exploiting the resources distributed on networks. Their main inadequacy is that they do not currently provide support for legacy programs written in other languages such as C and Fortran. They are limited to distributing computation-intensive applets to Web clients or servers and collecting and organizing the results.

FAFNER is a set of CGI scripts designed to support automated task distribution, relation collection, and archival services, password-controlled anonymous user database administration, and hierarchical multi-server configuration for the sieving phase of a global RSA factoring project [CRC 96]. It uses the WWW to distribute parts of a large problem and for coordinating a large number of clients that take over a part of the overall computation. The participants sends the partial results back to the main FAFNER server at regular intervals. Each host may participate in and take role in the solution of the global operation by activating a FAFNER Web server. This servers collaborate to manage the distributed task queue, coordinate client processes running within the server's domain, gather and unify results, and store and propagate compressed archives of partial relation data to the site that will perform the final reduction. Since a participant executes real binary files using its resources, the operations in the FAFNER environment depends on a mutual trust relationship.

ATLAS [BBB 96] is an incremental approach to building a metacomputing infrastructure. It uses existing technologies and builds on previous work. It manipulates a tree hierarchy to ensure scalability and uses Java and a URL-based file system.

ParaWeb [BSS+ 96] was developed at York University. It is a software infrastructure that supports the execution of applications written in a parallel version of Java that can be executed on resources distributed on the Internet or on a local Intranet. This Java dialect was developed by adding new classes to extend the Java programming environment and runtime and by exploiting its multithreading facilities. A special server called a *scheduling server* registers a set of compute servers ready to execute users' programs. The user of the system obtains the addresses of a number of compute servers by contacting the scheduling server and launches threads of the code on those compute servers. When the operation is complete, the user must inform the scheduling server that the compute servers are available for future requests.

WebFlow [FoxF96b] is a proposal for building a Web-based visual program environment for dataflow-oriented, coarse-grained distributed computing. It proposes to use a mesh of Java Web servers as a control and coordination engine, and a channel-connected dataflow mechanism among coarse-grained Java modules.

The MetaWeb, Metacomputer Management Package  is a joint-project by Northeast Parallel Architectures Center at Syracuse University and Cornell Theory Center (CTC) that plans to build a fully functional graphical user interface to manage CTC’s EASY-LL task scheduler. EASY-LL includes two pieces of software: EASY (the Extensible Argonne Scheduling System) is a resource scheduler for homogenous cluster computing. LL (Load Leveler) is a resouce management system for networks of heterogeneous workstations and MPPs.

SuperWeb [AIS 97] is a Web-based global computing infrastructure based on three fundamental participants: brokers, clients, and hosts. Brokers collect and monitor resources and honor requests for the use of these registered resources. Hosts offer their computational resources for public use by registering themselves with a broker. Clients are users who need extra computational resources. They send their requests to SuperWeb brokers and the brokers find a set of suitable hosts that are capable of serving the clients’ requests. Serial or parallel computational tasks written as Java applets are launched on registered hosts. The responsibility of writing a flexible program (i.e., Java Applet) that can coordinate the activities of other Java applets distributed to hosts to perform a common task belongs to the clients.

Charlotte [BKK+ 96] supports a programming model that provides an abstraction of infinitely many processes sharing a common name space. Programs are written using standard Java with parallel tasks specified as Charlotte routines. A routine is analogous to a standard Java thread with the exception that it is capable of being executed remotely. The first process starting to execute the given program becomes the *manager process*. When a routine is reached, the manager determines the number of processes (*workers*) that is required by the parallel step and waits for workers to contact it. It is assumed that users who would like to donate their resources will connect to a SuperWeb broker URL address using a Web browser and indicate their intention. The number of donators at any one time should be equal to or greater than the number of workers required in order to be able to continue the execution of a parallel step. The manager waits for the workers to complete their task assignments before continuing to the next sequential step.

DAMPP (Distributed Applet-Based Massively Parallel Processing) [Vanh 97] is another Java-based system similar to Charlotte and SuperWeb. A *master process* divides the problem into many small tasks, launchs the tasks on connecting clients' computers as Java applets, and combines the results of the individual tasks to get the final result.

All three systems, namely SuperWeb, Charlotte, and DAMPP, are guided by the same principles. They all assume that users connect to a central server site using their browsers and volunteer to execute some computational tasks. Clients need to pull out the jobs since the server cannot push the tasks to individual sites when required; therefore, if not enough clients are available, execution of a job may be delayed indefinitely. Furthermore, there are some application-specific drawbacks to using such systems where tasks should be expressible as small programs and individual tasks cannot inter-communicate among each other. All the communication occurs between the server and an individual client. The use of Java applets as computational task entities is a further restriction since these tasks cannot perform input/output operations. All these drawbacks limit the target applications where such Java-based systems could be useful indeed.

## 1.6 Organization of the Thesis

The organization of the rest of this thesis is as follows: Chapter 2 presents the implementation of the World-Wide Virtual Machine based on CGI-extended HTTP servers and PVM communication daemons. The same chapter also describes the details of the coordination engine of the WWVM that give it the capability of running as a dataflow-based global computing machine.

Chapter 3 describes implementation issues such as building graphical user interfaces for the Web, configuration and extension of the Web servers that constitute the base of the WWVM architecture, mechanisms to provide users with system accounts, and the pros and cons of using X-window-based, server-side facilities or client-side helper applications versus implementing compatible facilities from scratch using client-site Web technologies such as HTML, JavaScript, Java, and plug-in extensions. An in-depth discussion of the WWVM's security mechanisms is also given.

Chapter 4 presents the shared-memory and message-passing programming models supported by the WWVM, and the porting of HPCC software tools onto the WWVM platform.

The role of the Java language in providing platform-independent visualization software components is emphasized in Chapter 5. The design of the WWVM performance and data visualization components is described in detail, and implementation of data wrappers follows.

Chapter 6 summarizes the work done in this thesis, highlights major contributions, and presents future directions and possible extensions to the presented research.