WebWork: Integrated Programming Environment Tools for National and Grand Challenges

Geoffrey C. Fox

Department of Computer Science, Department of Physics and
Northeast Parallel Architectures Center, Syracuse University, Syracuse, NY
3154432163 FAX 3154434741 gcf@npac.syr.edu
http://www.npac.syr.edu or http://www.infomall.org

Wojtek Furmanski

Department of Physics and
Northeast Parallel Architectures Center, Syracuse University, Syracuse, NY
(315) 443-3891 fax (315) 443-9103 furm@npac.syr.edu http://www.npac.syr.edu

Marina Chen

Department of Computer Science, Boston University, Boston, MA
(617) 353-8919 fax (617) 353-6457 mcchen@cs.bu.edu http://cs-www.bu.edu

Claudio Rebbi

Department of Physics and Center for Computational Science, Boston University, Boston, MA
(617) 353-9059

James H. Cowie

Cooperating Systems Corporation, Chestnut Hill, MA
(603) 464-5799 Fax:(603) 464-6014 cowie@cooperate.com http://www.cooperate.com

Joint Boston-CSC-NPAC Project Plan to Develop WebWork
June 14, 1995

1 Project Summary

Challenging-scale problems consistently demand solutions that fuse geographically distributed and heterogeneous data, personnel, expertise, and resources. For example, national health care problems require collaboration among experts of fields as diverse as medical informatics, public policy, robotics, and high performance computing to solve problems ranging from telemedicine to cost management and quality control. Indeed, many National Challenges include Grand Challenges as subcomponents. We can pose the Integrated Challenge as the solution of metaproblems hosted on world-wide metacomputers linking all three aspects: simulation, information processing, and collaboration.

We suggest a hybrid approach to Integrated Challenges that combines World-Wide Web (WWW) technologies with the current portable scalable software systems developed by the HPCC Initiative for Engineering and Scientific simulations. This combines the parallel languages and runtime of HPCC with the software engineering, collaboration and pervasive technology base of the WWW. The resultant system will support collaborative rapid prototyping, applied to problem solving environments (PSE) at different scales. Support will extend from systems such as ELLPACK (parallel partial differential equation PSE), NWChem (parallel computational chemistry PSE), MATLAB (uniprocessor matrix and signal processing PSE), or Khoros (image processing/visualization PSE) in the simulation arena to complex National Challenges including education, manufacturing and healthcare. Roughly we consider the world-wide metacomputer as a set of future generation WWW servers, each of which (individual workstation, cluster, SMP, or MPP) is implemented internally (in the closely-coupled homogeneous environment) using appropriate HPCC technologies (e.g. HPF, MPI, PVM, Fortran-M, pC++, CC++, parallel Oracle, etc.). Then we link these servers together using generalized WWW technologies to allow executable program components to be published as services, and so create a distributed problem solving environment for prototyping large scale software.

We therefore propose a careful three layer design and prototype implementation of WebWork, an extensible world-wide virtual machine (WWVM) model for heterogeneous and distributed high performance computation. In our three year plan, we propose to initially develop the basic software engineering tools in a bootstrap fashion – our first WebWork tools will accelerate the distributed development of further capabilities. Note that whereas MPP's have rather limited development environments, the World Wide Web is likely to have the best productivity tools of any system. Basing HPCC on Web technologies allows us to leverage these tools to produce a powerful software engineering environment. We will prototype the WebWork computational services and test them on several applications – including both Grand and National Challenges for which prototypes have already been developed using conventional methods and for which metacomputers and HPCC are appropriate approaches. In particular, we will develop tools to support computer centers set up as distributed metacomputers. This will be tested on a modest scale at Boston's Center for Computational Science and Syracuse's NPAC but designed for applicability to the NSF Metacenter.

2 General Principles and Motivation

2.1 Motivation Challenging-scale problems consistently demand solutions that fuse geographically distributed and heterogeneous data, personnel, expertise, and resources. For example, national health care problems require collaboration among experts of fields as diverse as medical informatics, public policy, robotics, and high performance computing to solve problems ranging from telemedicine to cost management and quality control. Another example is the manufacturing domain: aircraft are now built by virtual corporations with international scope involving several large and thousands of small companies. The computational challenge involves integrating tens of thousands of programs and expert systems with distributed collaboration. We can pose such an Integrated Challenge as the solution of metaproblems hosted on world-wide metacomputers linking all three aspects: simulation, information processing, and collaboration. As many National Challenges include Grand Challenges as subcomponents, we suggest a hybrid approach to Integrated Challenges that combines World-Wide Web (WWW) technologies with the current portable scalable software systems developed by the HPCC Initiative for Engineering and Scientific simulations. This approach naturally links computation with the evolving but already excellent collaboration support of the Web. Compared to MPP's, known for their primitive environments with poor or nonexistent productivity tools, the Web is rapidly becoming the best supported computing environment with a remarkable set of public domain and commercial tools. All these factors make the Web a natural basis for an HPCC software engineering environment, built from a bootstrap of Web-based high performance computing systems.

We therefore propose *WebWork*, a collaboratory multi-user multi-server problem solving environment on the Internet, based on the existing HPCC technologies for local computational backends and the evolving WWW technologies for human interfaces, system-wide coordination and world-wide dissemination. It supports collaborative rapid prototyping, applied to programming environments at different scales ranging from support from a single PC running a WWW server and client, to the enterprise computing systems of a corporation or a university department, to the NSF Metacenter and even multi-national, world-wide collaborations. WebWork's view of the world-wide metacomputer is a set of future generation WWW servers, each of which (individual workstation, cluster, SMP, or MPP) is implemented internally (in the closely-coupled homogeneous environment) using appropriate HPCC technologies (HPF, MPI, PVM, Fortran-M, pC++, parallel Oracle, etc.). Then we link these servers together using generalized WWW technologies to allow executable program components to be published as services, and so create a distributed problem solving environment for prototyping large scale software.

2.2 General Principles WebWork computation is built around the concept of a *compute-server* – an evolving Web technology server – as a base processing unit. Already today, Web servers such as NCSA HTTPD can spawn arbitrary computational *modules* via the CGI protocol, and they can be linked via HTTP message passing based *channels* to form a distributed processing system that we call the *World-Wide Virtual Machine* (WWVM). Furthermore, the exploding installation base of Web servers, a gradual blend of client and server technologies (e.g. Java interpreter), and their increasing computational (e.g. Netscape) and communication (e.g. HTTP-NG) capabilities, will result in continuous performance increase of such a WWVM. Finally, the software industries spawned by the Web expansion guarantee continual improvement in the quality of the underlying Web software infrastructure. WebWork will harness this unprecedented performance and quality to build a new generation of *Problem Solving Environments* for Grand and National Challenges.

Another key concept of the WebWork model is the *Virtual Software Laboratory* (VSL) – a software engineering framework based on *Web Productivity Tools* (navigators, databases, editors), enabling collaborative *software processes* in which WebWork bootstraps itself by being used to build its own software. WebWork applications are specified as networks of linked modules called *compute-webs* and organized according to the dataflow paradigm, popularized in the scientific community by systems such as AVS or Khoros, and here termed *WebFlow*. The model integrates publication and computation by offering WebWork software *publication* standards across disciplines, institutions and geographic locations. Web HPC developers will publish both computational *problems* (compute-webs with missing components) and *solutions* (modules), and employ *agents* to perform suitable matching.

2.3 Three Layered Architecture WebWork is a three layer software architecture. Layer 1 is an extensible World-Wide Virtual Machine (WWVM) for heterogeneous and distributed high performance computing; it defines computation and communication primitives, initially based on existing Web standards (MIME, HTTP, CGI, CCI), to provide a publication model of computation. Layer 3 contains the user level programming and problem-solving environments (PSE's). These domain-specific PSE's range from NWChem and Global Arrays for computational chemistry at Pacific Northwest laboratory, ELLPACK (parallel numerics PSE), MATLAB (uniprocessor PSE), and Khoros (signal processing/visualization PSE) in the simulation arena, to complex National Challenges including education, manufacturing and healthcare. This minimal two-layer model supports only traditional forms of client-server interaction; that is, computation is performed by a web of servers, while client tools act as simple presenters for the published output of the computation. As WebWork grows, rather than building PSE's directly atop the WWVM, Layer 2, or *middleware*, will add more advanced client paradigms to this basis. In Layer 2, advanced client codes called *agents* may take on server functionality, becoming autonomous participants in the compute-web. Increases in client-code mobility and flexibility will also require the addition of "brokers" to mediate the interchange of different data formats and "little languages" (VRML for 3D graphics, PERL for text processing, HPFCL for coarse-grained control-parallel HPF modules, AVS for visualization and dataflow, ELLPACK/PDELAB for PDE specifications, and so forth) between participants in a compute-web. This interpolating environment, *WebScript*, will be an evolving collection of competing approaches with agent based "translators" to implement interoperability.

2.4 Related Work WebWork does not compete with existing approaches; it integrates them by transferring HPCC languages, with their ability to express parallelism and their sophisticated runtime environments, to the Web. WebWork takes advantage of existing high performance tools (optimizing compilers) and runtime environments (low-latency messaging within a tightly-coupled MPP) within each individual unit of a metacomputer. Simultaneously, it takes advantage of the fact that the coordination of these coarse-grained program chunks, spread over the individual units of the metacomputer, may be better served by the flexibility and (emergent) robustness of interpreted Web environments. WebWork brings to HPCC the features of modern distributed computing: real-time systems, network security, (software) configuration control for distributed databases and more general CASE and software engineering support.

The evolution of HPCC, from single-site technologies to a group-based collaborative effort over the NII, will be critical if high-performance technologies are to have their broadest impact. The World-Wide Web, with support for collective publication and multisite collaboration, is rapidly emerging as the dominant arena for collaborative, group-oriented software development. The coordination and publication of HPCC technologies over the Web has therefore become an inevitable and critical step in the widespread acceptance of those technologies. Often one considers computing as a pyramid, with personal systems at the bottom and MPP supercomputers at the apex of the pyramid. Rather than directly porting HPCC technologies to the base, WebWork takes advantage of pervasive Web support at the base and builds to the apex, guided by the many key correct principles and systems produced by the HPCC Initiative.

2.5 The Proposal In the technical discussion that follows, we initially describe with a table and a succinct summary all the key subsystems and ideas in WebWork. Then we describe our proposed technical approach, divided into the Virtual Software Laboratory, four key applications with which we will test our designs and implementations, and then our proposed activities in the three layers described above. In the final section, we summarize both the deliverables and our team, which combines the computer science and applications expertise of three organizations.

3 Technical Background and Proposal

3.1 Project Scope and Terminology

WebWork introduces many new terms, and overloads some common terms with more specialized meanings for the sake of intuition. For example, we define "WebWork agents" (or, to be concise, simply "agents") as semiautonomous software modules that act as brokers between other software components. The following project summary collects all major terms and concepts of the proposed WebWork paradigm, marking our specialized vocabulary in italics. The attached table collects all terms and offers brief self-referential definitions. Apparent circularities are to be resolved by the WebWork boostrapping processes discussed in the proposal body.

3.1.1 Project Scope The goal of this project is to prototype and demonstrate the *WebWork* [1.1] based *Problem Solving Environment* [1.2] for National and Grand Challenges. To accomplish this task, we will use today's *server* [2.4.3] technology and publicly available components of *Web Productivity Tools* [2.2.1] as exemplified by NPAC *WebTools* [2.2.3] to bootstrap the initial *software process* [2.2.2] that will prototype *Virtual Software Laboratory (VSL)* [2.2.4]. We will then use VSL to prototype *World-Wide Virtual Machine (WWVM)* [2.4], and then use both to prototype, test and demonstrate a set of National or Grand Challenge *applications* [2.3]. We will initially employ the *top-down process* [2.2.2] methodology which facilitates quick formation and *publication* [2.4.1] of coarse grain *modules* [2.4.3] using the existing Challenge software components. Such modules are then linked in terms of dataflow *channels* [2.4.3] to form *compute-webs* [2.4.1], accessible by *clients* [2.6]. Visual compute-web *editor* [2.6.4] will be prototyped in the early stage of the project and then used to publish both *problems* [1.2] and *solutions* [1.2] for the selected set of Challenge applications. This will provide framework for testing our concept of the *agents* [2.5.2] aided collaborative problem solving environment. We will start with the most intuitive *WebFlow* [2.6.2] paradigm in which users directly employ visual editors to build compute-web connectivity graphs. A few application domains will be addressed concurrently – initially telemedicine and chemistry – and simple instances of problems and solutions will be identified by cross-cutting these disciplines in terms of emergent WebWork standards. For example, telemedicine develops an image viewer for pathology workstation and chemistry uses it to visualize, debug and monitor compute-webs for distributed array processing. Such interoperability is achieved by using common standard formats for all *object types* [2.4.3] and *documents* [2.4.1] such as inter-modular connection *ports* [2.4.3] and communication *objects* [2.4.3]. In the next stage, we will extend this approach to other programming paradigms and we will address base HPCC support for intra-modular computation in terms of the *High Performance Fortran Coordination Language (HPFCL)* [2.6.3] model. Here we will be also able to initiate and test the *bottom–up process* [2.2.2] by importing HPCC runtime libraries to the WebWork framework. Aspects of such a process, aimed at building consistent and complete module libraries for selected domains, will be addressed throughout the project and we will attempt to formalize it in the HPCC domain in terms of the *WebScript* [2.5.3] protocol. Such an evolutionary common intermediate form would provide an integration platform for a growing collection of *middleware* [2.5] protocols and a powerful, high level programming language based framework for publishing problems and solutions. With HPFCL, we will initiate the process of building the HPCC layer on top of pervasive Web technologies, testing it in the specific application context, prototyping parallel I/O in terms of server *databases* [2.6.3], and collecting specification requirements for the next generation multithreaded object-oriented *compute-servers* [2.4.3], driven by the WebWork paradigm. A suite of diverse test applications, selected for this project, such as RSA factoring, telemedicine, chemistry, and metacomputing support, will allow us to test HPFCL and integrate it with several other high level programming/authoring paradigms, such as WebFlow, Consumer/Producer and Visual Compute-Web Editor.

3.1.2 WebWork Terms and Concepts

Agent	a *middleware* broker *module* that facilitates *WebWork* operation	2.5.2
Application	a *WWVM*–runnable *compute-web* and its *clients*	2.3

Bottom-Up Process	a *software process* that extracts reusable *modules* from applications	2.2.2
Channel	a communication line between two *ports* used to exchange *objects*	2.4.3
Client	a Web browser or *editor*	2.6
Compute-Server	evolving Web technology server, driven by *WebWork* computation	2.4.3
Compute-Web	a composite *module* given by a dataflow network of *modules* linked by *channels*	2.4.1
Database	a *server document* tree with atomicity, integrity and concurrency cotrol support	2.6.3
Document	Web-viewable instance of an *object*	2.4.1
Editor	a Web browser with enhanced *WebFlow* authoring functions	2.6.4
HPFCL (HP-Fickle)	coordination script and interface builder for HPF *modules*	2.6.3
Middleware	any *WebWork* module that is not a *client* or part of *WWVM*	2.5
Module	computational unit with specified I/O *ports* and CGI interface to a *server*	2.4.3
Object	an instance of *object type* used by *modules* as a communication unit	2.4.3
Object Type	Internet-public or *WebWork*-private MIME type	2.4.3
Port	a *channel* terminal with specified *object type*, published by a *module*	2.4.3
Problem	a published *compute-web* with missing *modules*	1.2
Problem Solving Environment	a *WebWork* enabled, *agents* aided collaborative process of matching *problems* with *solutions*	1.2
Publication	*WWVM*–runnable *module* with Web-published interface	2.4.1
Server	any Web server with *database* support or a *compute-server*	2.4.3
Software Process	a *VSL* based two-tier (*top-down*, *bottom-up*) *WebWork* SE process	2.2.2
Solution	a published *module* to be matched with a *problem*	1.2
Top-Down Process	a *software process* that encapsulates applications as *modules*	2.2.2
Virtual Software Laboratory	*Web Productivity Tools* based CASE tools that facilitate the *software process*	2.2.4
WebFlow	user level *WebWork* dataflow based *application* development environment	2.6.2
Web Productivity Tools	any Web software that facilitates *WebWork* authoring	2.2.1
WebScript	*WebWork* coordination and management language, the associated *software process* and *agents*	2.5.3
WebTools	an instance of *Web Productivity Tools*, developed at NPAC to bootstrap *VSL*	2.2.3
WebWork	hierarchical network of *applications* and the associated *software process*	1.1
World-Wide Virtual Machine	*WebWork* infrastructure layer given by an interactive surface of interconnected *servers*	2.4

3.2 Virtual Software Laboratory

3.2.1 Web Productivity Tools Web technologies offer a viable standardization, publication and productivity platform for NII software engineering. Current *Web Productivity Tools* include free, low cost or bundled Web browsers such as NCSA Mosaic, Netscape and Internet Explorer, quality HTML editors such as HotMetal, Internet Assistant and WebMagic, database interfaces, and early collaboration tools. New generation tools, including interactive Web extensions such as Java [Java95] and VRML [VRML95] will offer rich functionality in the area of text processing, media authoring and GUIs.

3.2.2 WebWork Software Process As part of the WebWork project, we will develop *Virtual Software Laboratory* – a Web Productivity Tools based CASE environment that will enable WebWork *software process*. VSL tools will allow to cast existing software entities to a module format (we call it *top–down process*), or to develop reusable module libraries (we call it *bottom–up process*), and to publish it on the Web. Standardized publication modes will range from static exposure, including documentation and download instructions, to an interactive demo mode, to the full runtime service mode. In the latter, full *publication* mode, a module is ready to be plugged into any *compute-web* and directly participate in distributed WebWork computing.

VSL as specified above is a major effort by itself; it will require the involvement of the whole Web community to succeed, resulting eventually in a shopping mall of standardized module warehouses and compute-web boutiques for the NII. Within this project, we will propose initial standard candidates and provide minimal but operational experimentation framework to facilitate collective interactive exploration of alternative solutions and extensions of the model.

3.2.3 NPAC WebTools Input for the WebWork/VSL bootstrap will be provided by *WebTools*, currently in prototype at NPAC. WebTools [FoFu95] – a CGI-extended Web server accessible by any standard browser – offers tools for groupware software development (Web Editor), MIME database model for the server document tree (HyperWorld Manager), disciplined navigation through the source code and documentation (HyperWorld Navigator), early CASE tools for integrated development of hyperlinked source and documentation volume (HyPerl World), and collaborative e-mail support (WebMail). Currently, WebTools already contains the base Web software support required to build an embryonic WebWork. Individual tools will be adapted to WebWork requirements over the summer '95 and the customized WebTools package will be made available on day one of the project.

3.2.4 WebWork Bootstrap An embryonic WebWork will be constructed at the beginning of Year 1 by installing instances of WebTools server at Boston, CSC and NPAC. This will bootstrap the initial 3-node top level compute-web, including machines ranging from workstations, SMP, and MPP to parallel database and visualization systems. HyperWorld Manager will offer the initial database support for the WWVM operation, and HyPerl World CASE tools will support initial module prototyping and publication formats. NPAC is responsible for the WebTools delivery as specified above. Boston and NPAC will start application prototyping work in Year 1 that will enlarge the core WebWork by domain-specific compute-webs, and CSC will provide software engineering expertise in evaluating the embryonic WebWork, suggesting improvements, re-designing and re-implementing its infrastructure components. By the end of year 1, the WebWork bootstrap process will be completed, WebWork concept will be proposed in an Internet Draft, and the early WebWork software (WWVM, VSL and initial application prototypes) will be published on the Internet to encourage formation of new WebWork nodes, and to start collaborative refinement of the WebWork publication and computation standards. VSL will evolve accordingly and consolidate tools for development and management of application modules (Year 2) and reusable module libraries (Year 3).

3.3 Test Applications

WebWork's design and testing requirements call for a suite of sample applications. We consider a broad space of applications for the WebWork design but realistically must choose some relatively simple examples to implement for test purposes. After some general remarks, we describe four application areas - RSA Factoring, Computational Chemistry, Telemedicine and Metacomputing Centers - where we will apply and thoroughly test WebWork. Other areas (manufacturing, differential equation solver systems) will be used in WebWork design but no major implementations are included in this proposal. We have always worked closely with applications as illustrated by Boston University's Center for Computational Science where scientists from about twenty different departments and centers have been engaged in over 100 different research projects, making BU an acknowledged pioneer in the use of high performance computing for science and engineering applications. In addition, the recent book "Parallel Computing Works" [FMW94] describes over 40 applications developed at Caltech in group involving Fox (as PI) and Furmanski before their move to Syracuse. So our WebWork design certainly benefits from many years of experience in analyzing the software and HPCC requirements of large scale simulations. We note that several of the National Challenges combine major computing and information requirements and so are natural candidates for WebWork.

3.3.1 RSA Factoring We will initially test our two-layer prototype with an interesting "embarrassingly parallel" application involving the factoring of large (512 bit) numbers used in RSA coding. We are collaborating with the Bellcore team that pioneered the factoring by email approach to smaller problems of this type. We estimate this will need about one Teraop month to complete and we expect the publication model of computation to attract web users and so harness the many thousands of workstation/Pentium PC class machines needed to address it.

3.3.2 Metacenter Support The publication model also lends itself to support for batch computing and metered HPC services and is especially suited for the growing trend to geographically distributed metacomputer centers such as the NSF Metacenter. WWVM-published descriptions of available computational resources can ultimately include pricing and availability information. The WWVM layer can include client-side and server-side tools for tracking the costs incurred by the nodes of a given compute-web, and even the exchange of digital cash for immediate on-line payment. More importantly, however, WWVM batch-mode computing can help the providers of HPC resources manage access to and scheduling of those resources, improving availability and minimizing idle time. This project will be a major responsibility of BU and after testing at the Computational Science Center and NPAC (Syracuse) in Year 2, we will explore in Year 3 the use of WebWork in the NSF Metacenter.

3.3.3 Telemedicine Our second test application for WebWork involves a simplified healthcare (a National Challenge) problem which we have developed at NPAC with the local SUNY Health Science Center which has a nationally recognized telemedicine activity. Here we use networked workstations and an MPP (Maspar) to support image processing and data management for a "pathology workstation" where the surgeon has a computer controlled image capture device. We are currently implementing this with web-interfaced Oracle database and AVS image processing modules. We will use WebWork to implement a full web technology solution using the WebFlow layer 3 artifact described in Section 2.6.2.

3.3.4 Chemistry Our final examples are rather different, based on the NWChem approach to parallelization of chemistry problems developed at DoE's Pacific Northwest Laboratory. NWChem uses a "Global Array" (GA) abstraction [NHL94], and has been successfully applied to several important chemistry problems. GA's currently support any application built on top of full matrix data structures and both Boston and NPAC have much experience with networked solutions and compiler/runtime optimization for this problem class in the case of Computational Electromagnetics using the method of moments [CLFMH93]. This problem class can be naively broken into matrix generation (embarrassingly parallel) followed by matrix manipulation with eigenvalues, equation solution and multiplication important in different cases. Much of the matrix manipulation is coarse grain and so suitable for loosely coupled network implementation. As mentioned in Section 2.6, we will develop WebChem as an example of a domain specific problem solving environment. We will in the last year, link this to a parallel I/O server (Layer 1 in Section 2.4, see also Section 2.6.3) that we are building using web technology as this will explore - albeit in a simplified case - the critical link of databases (I/O) and computing in WebWork. Chemistry problems traditionally have major data management and I/O needs and also require compiler and runtime system expertise offered by this team.

3.3.5 Next Steps Manufacturing is a particularly rich example where NPAC is working with the MADIC (Multidisciplinary Analysis and Design Industry Consortium) collaboration involving Rockwell, Northrop Grumman Vought, McDonnell Douglas, General Electric, General Motors and Georgia Tech. Fox is in particular involved (and is project PI) in establishing for this NASA project, the NII requirements for a future concurrent engineering concept called ASOP (Affordable Systems Optimization Process). Aircraft are now built by multi-company collaborations with international scope; these virtual corporations need the collaborative and networked software engineering and workflow support built into VSL. Configuration management is a critical need. ASOP links a range of disciplines (from manufacturing process simulation, electromagnetic signature, aeronautics and propulsion computation linked to CAD databases and virtual reality visualization) using MDO – multidisciplinary optimization – techniques. The integration of conceptual (initial) and detailed design with the manufacturing and life cycle support phases naturally requires the integration of information and computing in WebWork. Note that the detailed design phase can involve the linking of thousands of separate (Fortran) programs in the optimization process – a great place for HPFCL at Layer 3 (Sec. 2.6.3). Each discipline has its own approach and interfaces which naively implies N^2 translation programs for N disciplines. This is partially addressed by the Industry Product Description Standard PDES/STEP specification where we translate all systems to a single intermediate form and so only require 2N translators. If our vision is correct, VRML and STEP will be merged to a suitable Web based standard data structure. Further each discipline is now essentially associated with its own interpreted little language which needs integration as in our proposed Layer 2 (Sec 2.5). We will continue to work with ASOP whose requirements document will be delivered September 95 and refine its implications for WebWork. Funds outside this proposal will be sought for exploring the use of Web technology in ASOP demonstrations.

Note that ASOP includes as a subset the solution of many different partial differential equations. Even by itself, this subproblem represents a good opportunity for WebWork — for example, as the software backplane for Purdue's PDELAB and the older (parallel) ELLPACK. Here we see a little language for specifying partial differential equations and Web technology (HPFCL) used both for the many different modules and internally to a single solver to control the different domains in modern domain decomposition or adaptive multigrid methods. NPAC is developing a toolkit with Texas as part of a numerical relativity grand challenge which could use WebWork for software integration.

We have described a rich application (manufacturing and partial differential equation solution) that motivates and provides requirements for most features of WebWork. We have also discussed a set of simpler applications which seem realistic to build using WebWork within the requested budget. Naturally we will review the broad requirements and specific implementations as technology and our understanding of applications evolve, so that we can modify our plans if needed.

We now proceed with a more detailed technical presentation of individual layers in the three-layered architecture of the WebWork system, followed by the project implementation plan.

3.4 Layer 1: WWVM – The World-Wide Virtual Machine

The WWVM is the distributed software and hardware basis for a *publication*–based model of computation, in which existing Web servers collaborate to provide simple, coarse-grained dataflow among Web-published software modules. The core technologies of this basis are the existing standards for client/server interaction and data exchange that are now in common use (CGI, HTTP, HTML, MIME). By defining the primitive mechanisms of publication-based distributed computation, WWVM provides the bootstrap for higher-level computational services of Layers 2 ("Middleware") and 3 ("Application Environments").

3.4.1 Distributed Computation as Collective Publication The WWVM takes advantage of the observation that a Web server can easily take the role of a Web client, requesting copies of information from other Web servers in addition to simply maintaining hyperlinks into their document space. In our initial release of the WWVM, *publication* will enable computation in two ways. To establish a simple, coarse-grained dataflow pattern among nodes in this graph, or *compute-web*, servers will publish descriptions of the software modules they offer, plus their "wish list" of unsolved problems. Once the user has constructed a compute-web, executing modules at each site will use the URL *document* space as an I/O scratch pad, publishing snapshots of internal state as input to remote computational modules as the computation progresses.

3.4.2 Integrating Current Approaches to Parallelism The WWVM's support for effective distributed computing is due primarily to its use of the best underlying Web server and client technology — whatever that may mean at any given time. For example, each of the large computational services published by a single node may itself be a PVM process internally, funneling the combined efforts of an organization-wide computational cluster through a single point of publication in the Web. Because the initial WWVM takes advantage of existing Web servers, and simple POST/GET requests from the HTTP protocol, users will be free to augment the WWVM with direct server-server message-passing protocols of their own choosing. From the start, this will make the WWVM an ideal environment from which to launch and monitor PVM computations.

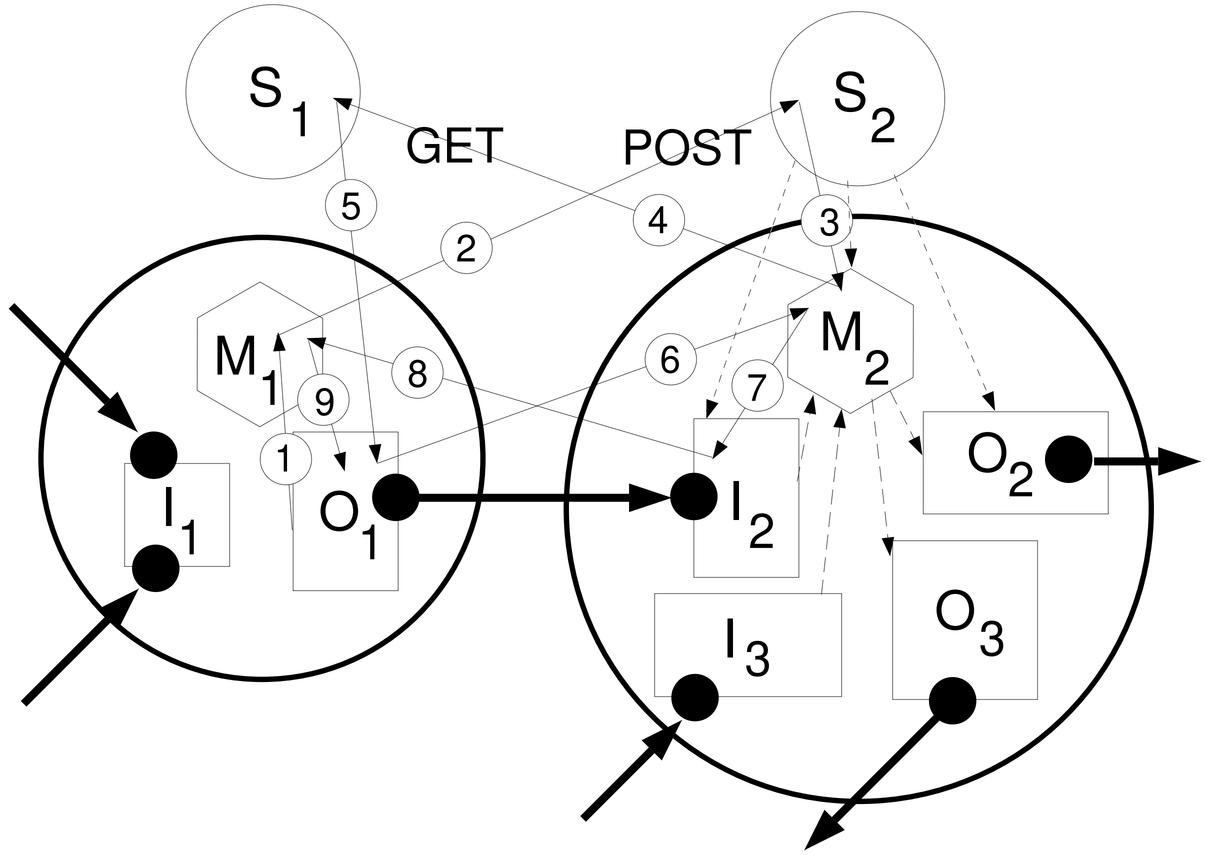

Figure 1 This diagram illustrates point-to-point communication between Web servers, used to implement a webflow channel between compute-web modules. Two extreme implementation modes are described: a) based on today's Web server technology, and b) based on thread memory mapped high performance implementation, expected in future Web compute-servers. Subsequent steps, represented by a sequence of labelled lines in the figure, are described below in both implementation modes.

a) Today's Web server mode: (1) – M_1 locks O_1 on S_1 disk. (2) – M_1 sends POST HTTP message to S_2 with M_2 URL in the header sector and with O_1 URL in the body sector. (3) – S_2 activates M_2 via CGI and passes O_1 URL as a command-line argument. (4) – M_2 sends GET method to S_1 with O_1 URL in the header. (5) – S_1 fetches O_1 from its document tree. (6) – S_1 sends the content of O_1 to M_2 which completes the GET exchange. (7) – M_2 saves O_1 by overwriting current I_2 on the S_2 disk. If I_2 is locked, M_2 waits (blocks). (8) – After O_1 is saved on the S_2 disk, M_2 returns 'end-of-transfer' acknowledgment to M_1 which completes the POST exchange. (9) – M_1 unlocks O_1 and exists.

b) Compute-server (future Web server) mode: (1) - M_1 locks its memory object O_1. (2) - M_1 checks if socket connection to M_2 is in M_1 connection table. If yes, go to (5) below. Otherwise, M_1 connects to S_2 and sends M_2 creation script. (3) - S_2 spawns M_2 and acknowledges. (4) - M_1 receives acknowledge message and saves new socket in connection table. (5) - M_1 gets O_1 handle. (6) - M_1 writes O_1 to M_2 using socket lib calls. (7) - M_2 reads O_1 using socket lib calls. If I_2 is free, O_1 buffer is copied directly to I_2 buffer. If I_2 is locked, M_2 creates O_1 clone and blocks. (8) - M_2 sends acknowledge to M_1. (9) - M_1 unlocks O_1 and blocks.

3.4.3 Sample Implementation We summarize our discussion of the WWVM by presenting a sample implementation of publication-based dataflow primitives. Figure 1 illustrate an example of our Year 1 and Year 2 releases of WWVM; two Web servers (S_1, S_2) are engaged in building the dataflow communication channel between two modules (M_1, M_2). Heavy lines (circles, segments, dots) represent the WebWork dataflow metaphor (modules, channels, ports); thin lines illustrate the implementation in terms of existing Web technologies. Hexagons represent CGI modules, while rectangles represent input and output MIME files published in the document trees served by S_1 and S_2.

The particular action exposed in Figure 1a is an object transfer along the channel connecting modules M_1 and M_2. Module M_1 receives I_1 as input and produces O_1 as output. Module M_2 receives I_2 and I_3 as input and produces O_2 and O_3 as output. Objects O_1 and I_2 are connected via dataflow channel, and use a common MIME format. M_1 has just produced O_1, installed it as a MIME document in the document tree served by S_1, and notified M_2 of the change in its input, using HTTP protocol methods POST (to pass the address) and GET (to pass the data).

Figure 1b describes a more sophisticated implementation of WWVM dataflow; it shows how current Web servers will likely evolve with the advent of multithreaded and object-oriented text transfer protocols. The stateless HTTP based and disk I/O intensive implementation of the WWVM dataflow will result in high latencies and will be most appropriate for coarse grain modules; advances in these underlying protocols will translate into improved WWVM computational performance, thanks to memory object caches, thread based modules, and high performance server-to-server communication.

3.4.4 Layer 1 Bootstrap By the end of year 1 we will publish the initial version of the WWVM, in the form of a CGI script package to upgrade existing Web servers "in the field." This software will enable any collection of existing Web servers, using nothing more complex than the existing Hypertext Transfer Protocol, to act collectively as a simple graph-structured computational surface. This release, together with the software engineering tools provided by the initial Virtual Software Laboratory, will let users publish existing C++ and Fortran codes as Web services, and construct compute-webs from a collection of published services using coarse-grained dataflow. Installed at NPAC, BU, and CSC, this software basis will form an initial WWVM testbed, integrating PVM-based clusters of workstations with individual high performance machines for our first Web-based applications.

In years 2 and 3, we will release upgraded versions of the WWVM to assimilate advances in Web server technology and protocols. WWVM's generality and performance will improve as we address related subproblems from distributed computing. These extensions (atomic primitives, distributed timekeeping, and security provisions) will be closely analogous to existing solutions; for example, a set of atomic primitives in support of multiple-writer Web pages might include atomic page exchange, migration of single-instance pages from server to server, and the ability to lock multiple distributed Web pages while they are operated on as a group.

By deliberately including only the most basic published solutions of each of these problems, the WWVM will help bootstrap the development of more sophisticated treatments (hard realtime, hard security) of Web computation by existing academic and commercial research efforts in those specific fields, while avoiding direct competition and reinvention.

3.5 Layer 2: Middleware Integration

Simple WebWork applications, such as *Consumer-Producer* discussed in Section 3.6.1 can be easily constructed using the standard two-layer client-server architecture. In general, however, we will also need middleware support to assure interoperability and fault tolerance of the WebWork operation.

3.5.1 Complexity The software industry currently offers several competing *middleware* standards for client/server computing, including OpenDoc, Lotus Notes, SQL databases, TP monitors, groupware, agents, workflow and distributed objects (CORBA, OLE). The overall picture becomes increasingly complex and still far from full open standards based interoperability. The corresponding Web middleware, initially represented by simple rapid prototyping languages such as Perl or Tcl, has become similarly multi-faceted, including advanced programming models such as Java and VRML (Virtual Reality Modeling Language). Furthermore, advanced applications developed by National and Grand Challenges often come with their own, domain specific "little languages" which contribute to the proliferation of middleware. Finally, the HPCC domain brings to the table a suite of protocols such as PVM/MPI, programming models such as HPF and HPC++, and runtime environments such as Nexus or PCRC libraries, all of which which need to be integrated with pervasive and/or industry standard middleware.

To carry out the WebWork pilot project, we will need to face some aspects of this complex middleware integration problem. We plan to address it in two Layer 2 thrusts, based, respectively, on *agents* and a common intermediate form called *WebScript*.

3.5.2 Agents Initial WebWork tests in Year 1 will use simple 2–layer model in which Web browsers as clients interact directly with the WWVM servers. However, even the simplest user level WebWork environment, *WebFlow*, will require middleware layer as discussed in Section 2.6. Support for such intermediate entities, which we call *agents*, will be provided for specific application needs. The WebFlow agents mentioned above simply compensate for the mismatch between client and server programming models and/or data formats. We will develop suitable protocol translators from WebFlow to Java and VRML in Year 1, and from HPFCL to Java and VRML in Year 2. We will also address the issue of more advanced/intelligent agent support for load balancing, resource allocation, automated problem solving support, and CORBA/WebWork interoperability.

3.5.3 WebScript Advanced agents, such as those used in the Magic Cap/Telescript model, employ high level scripting protocols for transportation, replication, negotiation and information exchange. An elegant solution of the middleware integration problem could be constructed by organizing all interoperability and higher agents in terms of language structures of some common intermediate form, and then make both WebWork clients and servers compatible with such a language. Telescript's multi-server model, where the Magic Cap client is also a Telescript interpreter, already offers such a framework, and Java/HotJava could be naturally extended by developing Java interpreter based WebWork compute-servers.

We also explored previously a similar methodology in the MOVIE (Multitasking Object-oriented Visual Interactive Environment) [Furm94] server based HPF interpreter [FFHNS93] at NPAC, and developed an extended PostScript syntax based language, MovieScript, to integrate selected computational domains such as windowing (Motif), networking (sockets) and matrix algebra (Fortran90). MOVIE will be used in Year 3 of this project to package PCRC runtime libraries as WebFlow and HPFCL modules.

A concept of a common intermediate form for the WebWork model, called *WebScript* will be addressed in the second part of this project. Design of a common Web middleware must clearly involve the whole community. Our limited goal in this project is to formulate and publish the problem, and to evaluate several solution candidates mentioned above, based on the lessons learned in prototyping WebWork applications, as well as our previous experience with common intermediate language formats for multitarget optimizing compilers [CCW95] and interpreters [FFHNS93].

3.6 Layer 3: High Level Environments

WebWork users will be offered a collection of high level programming or GUI based *authoring* environments, suitable for assembling various application classes from module libraries. We discuss here a few such user level environments that will be prototyped as part of this project.

3.6.1 Consumer-Producer The simplest non-trivial compute-web contains a single channel linking 'consumer' and 'producer' modules. Such links will be often aggregated to form vertices with the 'host' module at the center, linked to a set of 'node' modules. Several popular groupware or/and collaborative settings can be naturally implemented in terms of suitable compute-webs, such as: a) *master-slave* based 'embarrassingly parallel' computation with a set of producer nodes returning asynchronously their partial results to the consumer host master; b) *project management* support that allows a consumer host to check quickly and at any time a collection of progress report pages in a set of producer nodes; c) *webcast* support, e.g. for distance learning, in which a producer host broadcasts the current document (viewgraph, video clip etc.) to a group of consumer nodes; or d) *webchat/teleconferencing* support, in which the current chatter/speaker becomes a producer host using a dynamic, transient webcast topology.

We will implement one computationally intensive consumer-producer WebWork application – the RSA factoring challenge, described in Section 2.3.1. The topology here is of the master-slave type, producers are volunteers that participate in the challenge, and consumer is the core WebWork described in Section 2.2.4 that collects the results. However, the eventual 'consumer' of this project's 'product' will be a customer of a bank whose security is RSA-based, who will be publicly informed if his/her bank code can or cannot be currently broken by the world's fastest computer – the WebWork.

3.6.2 WebFlow Simple consumer-producer type compute-webs can be easily authored in terms of today's Web browser technologies. Indeed, formation of a compute-web vertex requires only a list of participating nodes, which can be easily selected by the host user from WebWork 'yellow pages' in terms of conventional forms/widgets such as multi-selectable scrolled lists. For more complex graph topologies, including bi-directional or dynamic channels, hierarchical DAGs or closed loops, we need more flexible high level compute-web authoring tools. Dataflow visualization environments such as AVS [CFFFLM93] or Khoros offer a useful metaphor based on the notion of an active editable computational graph, with modules represented as labelled boxes or icons, and with channels represented as color coded lines, connecting appropriate modules. We will prototype two instances of such visual interface as part of the WebWork project and we discuss it in more detail in Section 2.6.4.

WebWork dataflow environment, or *WebFlow*, allows WebWork users to design, run, test and monitor coarse grain applications by directly assembling and interacting with the active visual graphs that represent compute-web topology. WebFlow implementation includes visual compute-web editor and agents which map between user level and WWVM level compute-web topologies. Early WebFlow prototype will be available by the end of Year 1, and then tested and refined in the context of metacenter, telemedicine and chemistry applications in Year 2 and 3.

All Layer 3 programming paradigms translate into Layer 1 dataflow, so that WebFlow maintains close correspondence between user-level and machine views of the WebWork operation. This graphical approach is most useful for a small collection of coarse grain modules with arbitrary irregular connection topology. On the other hand, symbolic techniques are more practical for large but regular compute-web authoring tasks such as setting a webcast (in which case the 'yellow pages' based selection is more efficient) or decomposing a massively parallel application.

3.6.3 HPFCL VSL will provide support for building WebWork modules in several popular sequential programming languages (C/C++, Fortran, Perl, Java). Full HPF support is beyond the scope of this pilot project, but we will initiate this process by designing and prototyping a minimal subset of encapsulated HPF called HPFCL (High Performance Fortran Coordination Language). HPFCL offers the base HPF support for building WebWork modules in terms of distribution/alignment directives, runtime library calls and MODULE and INTERFACE constructs. HPFCL delegates all number crunching to pre-compiled runtime library calls and focuses on handling module I/O, object layout and encapsulation. All these tasks can be implemented in the interpreted mode without major performance losses since high performance is guaranteed by the optimized data parallel runtime.

The runtime support will be provided by the Parallel Compiler Runtime Consortium (PCRC) [FRSM+93] [CoHa95] libraries, currently in the specification stage, and to be available in WebWork object and module formats in Year 3. Parallel I/O is at the moment still in the research stage and not planned as part of PCRC release. We will prototype parallel I/O support using WebWork document database constructs discussed in Section 2.2 and 2.4, and we will combine it with PCRC libraries as part of the HPFCL backend.

HPFCL interpreter will be designed, developed and integrated with WebFlow in Year 2, and then integrated with the PCRC, parallel I/O and the WebFlow editor, and used for building chemistry applications in Year 3. Top/application level coordination will be provided by the visual WebFlow paradigm. For selected chemistry objects such as Global Arrays, we will also start exploring the issue of WebScript based symbolic coordination. This will prepare infrastructure support for mapping large scale data parallel computation onto meshes of WWVM compute-servers which will be addressed in a follow-on project.

3.6.4 Visual Compute–Web Editor We will bootstrap in Year 1, prototype in Year 2, and refine in Year 3 a visual compute-web editor to support the WebFlow programming environment. The year 1 implementation will bootstrap the WebFlow model by reusing and adapting for WebWork purposes a publicly available dataflow environment such as Khoros. Khoros 'virtual' modules will play the role of WebWork agents, translating user visual design down to the WWVM layer. Khoros based WebFlow will be used to prototype chemistry applications in Year 1. In year 2, we will add HPCC support to this application domain in terms of selected HPFCL modules.

We will also prototype in Year 2 a Web browser embedded WebFlow editor, based on the best available interactive Web technologies. At the moment, the most promising candidate is the Java/HotJava model, recently published by Sun Microsystems and licensed by Netscape. In a Java based visual compute-web editor, WebWork modules will be represented as active applets that can adopt various visual dynamic forms depending on user preferences and ranging from labelled boxes to little simulations. When selected from a palette/database, dragged-and-dropped into the active editor area, and linked to the existing compute-web, they will instantiate suitable modules and connections at the WWVM level. Agents will be implemented as Java threads in the client domain which will enforce reliability and provide convenient platform for experiments with high level services such as fault tolerance, resource allocation, navigation through large hierarchical webs etc. WebFlow editor will be integrated in Year 3 with already existing backend support for the chemistry applications such as HPFCL modules and parallel I/O, and delivered as a proof-of-the-concept complete large scale WebWork application.

Support for URL-icon mapping and drag-and-drop based HTML editing is already offered by Internet tools integrated with Windows'95 (Internet Assistant, Internet Explorer) and the WebFlow editor will offer a natural extension towards and integration with computation. Active graph based authoring will soon become a popular metaphor for business computing, for example, in workflow tools for complex transaction processing; our visual WebFlow authoring tools will eventually be complemented by the next generation commercial desktop products.

4 Project Organization and Implementation

4.1 Teaming The PI's have actively discussed this project since the end of 1994; in that time, we have established an excellent collaboration which builds on our common expertise in HPCC and parallel compiler technology. As discussed in Sec. 2.1, the virtual software laboratory is designed to support geographically distributed software development, and we intend to test the VSL by utilizing it fully throughout the course of our project as described previously in Section 2.2.4 and below in Section 3.2. The team members are also within easy air travel range for conventional face to face meetings, so we expect this team to form a successful collaboration.

The team brings together expertise in a wide range of applications [PCW94] [BCLL92] and software [FFHNS93] at both message passing and high level (data parallel) compilers [CCW95] [BCFHR93] [ChCo92] [LiCh91] [LuCh91] for a range of HPCC architectures and approaches [FFHNS94] during the last 10-15 years. Further as described in the last section, we have developed substantial expertise in the World Wide Web with in particular an initial set of tools for software engineering, mail, file management and hyperworld navigation [FoFu95]. Further we have used and extended the Web technology for use on the New York state-funded Living Textbook [MFCM+95] project which links the parallel servers at NPAC via NYNET [DJWF+95] to six K-12 schools.

4.2 Collaboration Plan WebWork is a collaborative project that will develop and deliver tools for collaborative problem solving. We will apply similar bootstrap process as discussed previously in the software domain to establish an efficient collaboration space spanning NPAC, Boston and CSC. NPAC will deliver WebTools to the group, forming a core WebWork as described in Section 2.2.4 to be used initially to exchange and accumulate early software components and design elements. The goal of the first critical year of the project is to bootstrap the system infrastructure (WWVM, VSL), develop early application demonstrations, prototype the publication model and publish base WebWork to initiate Internet/Web discussion. Initially, each site will focus on one application task and all sites will collaborate on the infrastructure task, coordinated by CSC. NPAC will focus on telemedicine and Boston on Metacomputing support. After forming and testing an embryonic WebWork, two collaborative application tasks will be also initiated: Chemistry, shared by Boston and NPAC, and RSA factoring, shared by Boston, CSC and NPAC. After the first year, geographic location is expected to play a decreasing role and various shifts in task assignments will be natural and plausible. Detailed planning beyond the first year is difficult due to the following unknown factors: a) WebWork publication will generate feedback from the Web community and turn part of the core WebWork into an 'event driven' operation mode; b) early application demos will likely attract developers from individual domains. New infrastructure tasks for the core WebWork in Year 2 are: parallel I/O, middleware (e.g. Java) and HPFCL. Parallel I/O will be shared between CSC and NPAC, middleware between Boston and NPAC, and HPFCL between all three teams. Year 3 will be focused on integration and final delivery and the task assignment will be naturally determined by the expertise of individual teams, developed during Year 1 and 2.

4.3 Timelines and Deliverables

Year	Virtual Software Laboratory	Sec.
1	WebTools based VSL boostrap completed	2.2.4
2	Development and management tools for application modules	2.6.2
3	Development and management tools for reusable library modules	2.6.3
Year	World-Wide Virtual Machine	
1	Design and implementation of the NCSA HTTPD based WWVM	2.4.4
2	Support for parallel I/O	2.6.3
3	Selected compute-server extension tests (HTTP-NG, Java etc.)	2.4.3

Year	Middleware Integration	
1	Integrating Java and VRML with WebFlow	2.5.2
2	Integrating Java and VRML with HPFCL	2.5.2
3	Integrating PCRC runtime libraries with WebFlow and HPFCL	2.5.3
Year	High Level Environments	
1	WebFlow – module wrapper libraries, link with external editor (Khoros)	2.6.2
2	HPFCL interpreter, integrated with WebFlow	2.6.3
3	WebFlow and HPFCL integrated with WebWork Editor	2.6.4
Year	Application Demonstrations	
1	RSA Factoring, Chemistry and Telemedicine in WebFlow	2.3.1
2	Chemistry, Telemedicine in HPFCL, Metacomputing Tests at Boston and NPAC	3.2
3	Chemistry and Telemedicine with WebFlow Editor, Metacomputing in NSF Metacenter	3.2

4.4 Facilities

We can leverage substantial parallel computing and high-speed networking infrastructure [Podg94]. At Syracuse, this includes NYNET [DJWF+95] - the state-wide ATM network with double OC3 wide-area links. We are currently the major user of NYNET with a set of projects (Virtual Reality, Data transport, Collaboration and Video-On-Demand), funded by Rome Laboratory the State of New York and NYNEX. High Performance Computers at NPAC include a range of modest systems – SGI Challenge, IBM SP-2, CM-5 from Thinking Machines, nCUBE2 and a Maspar, plus a number of workstation clusters (DEC Alpha, SGI, IBM) supported by high speed networks (ATM LAN, switched FDDI). Most of these facilities are funded by the State of New York as part the InfoMall [FBJM+94] program. As part of NYNET activities, NPAC has integrated ATM and ISDN technology into its networking infrastructure which also includes the FDDI backbone and a direct T3 link to the Internet. Diversity of NPAC resources offers an excellent testbed for the metacomputing application of WebWork. Also note that NYNET links NPAC to the Cornell Theory Center and so to the NSF Metacenter infrastructure.

The computing environment at Boston University has continued to grow and now includes a 64 node CM-5, an 8 processor SGI Power Onyx compute/graphics server, numerous high performance workstations, an advanced graphics laboratory with animation and video production capabilities, HiPPI and FDDI high performance networks in addition to universal Ethernet connectivity and a 10Mb per second connection to the NSFnet backbone. Starting later this year and continuing through the second half of 1996, with support from a major NSF grant, BU will phase in our next generation supercomputing facility to replace the CM-5. Although we have not finalized the selection, we are targeting a 96 to 128 node machine with a peak performance in excess of 60 Gflops with at least 12 Gbytes of memory and 100 Gigabytes of high performance disk. The collection of facility clearly forms a metacomputer and would enable the testing of WebWork.

5 Glossary of Acronyms

ASOP – Affordable Systems Optimization Process
AVS – Application Visualization System
CAD – Computer Aided Design
CASE – Computer Aided Software Engineering
CCI – Common Client Interface
CFD – Computational Fluid Dynamics
CGI – Common Gateway Interface
CORBA – Common Object Request Broker Architecture
DAG – Directed Acyclic Graph
GA – Global Array
GUI – Graphical User Interface
HPC – High Performance Computing
HPCC – High Performance Computing and Communications
HPF – High Performance Fortran
HPFCL – High Performance Fortran Coordination Language
HPFI – High Performance Fortran Interpreter
HTML – HyperText Markup Language
HTTP – HyperText Transfer Protocol
MADIC – Multidisciplinary Analysis and Design Industry Consortium
MIME – Multipurpose Internet Mail Extension
MDO – Multi-Disciplinary Optimization
MOVIE – Multitasking Object-oriented Visual Interactive Environment
MPI – Message Passing Interface
MPP – Massively Parallel Processing
NII – National Information Infrastructure
NPAC – Northeast Parallel Architectures Center
OLE – Object Linking and Embedding
PCRC – Parallel Compiler Runtime Consortium
PDE – Partial Differential Equation
PDES/STEP – Industry Product Description Standard
PNL – Pacific Northwest Laboratory
PSE – Program Solving Environment
PVM – Parallel Virtual Machine
RPC – Remote Procedure Call
RSA – Rivest, Shamir and Adelman public cryptography algorithm
SGI – Silicon Graphics, Inc.
SMP – Symmetric Multiprocessing
SQL – Structured Query Language
SUNY – State University New York
TP – Transaction Processing
URL – Universal Resource Locator
VRML – Virtual Reality Modeling Language
VSL – Virtual Software Laboratory
WWVM – World-Wide Virtual Machine
WWW – World-Wide Web

6 Bibliography

[BCFHR93] Bozkus, Z., Choudhary, A., Fox, G., Haupt, T., and Ranka, S., “Compiling HPF for Distributed memory MIMD Computers”, in *The Interaction of Compilation Technology and Computer Architecture*, Kluver Academic Publishers, 1993. http://www.npac.syr.edu/HPFF/Fortran90D/book.ps

[BCLL92] Bhatt, S., Chen, M., Lin, C.Y., and Liu, P., “Abstractions for Parallel N-body Simulations”, Scalable High Performance Computing Conference (SHPCC ’92), Williamsburg, Virginia, April 1992. http://cs-www.bu.edu:80/faculty/mcchen/publication.html

[BGWSHC95] Beckman, P., Gannon, D., Winnicka, B., Sussman, A., Haupt, T., and Cowie, J., “PCRC Distributed Data Descriptors Specification – Draft 0.4”, PCRC Report, February 27, 1995. http://aldebaran.npac.syr.edu:1955/CCDDspec/ver0.4/DDD.ps

[CCW95] Chen, M., Cowie, J., and Wu, J., "CRAFT: A Framework for F90 Compiler Optimization." CPC95, 5th Workshop on Compilers for Parallel Computers. Malaga, Spain, June 1995, http://cooperate.com/docs.html

[CFFFLM93] Cheng, G., Faigle. C.., Fox, G.C., Furmanski, W., Li, B. and Mills, K., “Exploring AVS for HPDC Software Integration: Case Studies Towards Parallel Support for GIS”, in Proceedings of the 2nd Annual International AVS Conference *The Magic of Science: AVS’93*, Lake Buena Vista, Florida, May 1993. http://www.npac.syr.edu/PROJECTS/PUB/wojtek/hpsin/doc/avs93.ps

[ChCo92] M. Chen, M., and Cowie, J., "Prototyping Fortran-90 Compilers for Massively Parallel Machines," ACM Symposium on Programming Language Design and Implementation (PLDI), June 1992, San Francisco, California, http://cs-www.bu.edu:80/faculty/mcchen/publication.html

[CLFMH93] Cheng, G., Lu, Y., Fox, G.C., Mills, K., and Haupt, T., “An Interactive Remote Visualization Environment for an Electromagnetic Scattering Simulation on a High Performance Computing System”, Proceedings of Supercomputing ’93, Portland, Oregon, 1993. ftp://ftp.npac.syr.edu/AVS/electromagnetics.ps.Z

[CoHa95] Cowie, J., and Haupt, T., "Common Runtime Support for High Performance Fortran.", Parallel Compiler Runtime Consortium, April 1995. http://cooperate.com/PCRC/DDDF

[DJWF+95] McDermott, S.G., Johnston, W., Ware, B., Fox, G.C., Lunn, M.S., Kalos, M.H., Graniero, J.A., Davis, I., “The NYNET Upstate Corridor: A High-Performance Computing and Communications Network”, White Paper, Jan 1995. http://www.npac.syr.edu/users/hariri/publications/whitepaper/white.html

[FBJM+94] G.C.Fox, E.A.Bogucz, D.A.Jones, K.Mills, M.Podgorny and K.A.Hawick, “InfoMall: A Scalable Organisation for the Development of High Performance Computing and Communications Software and Systems”, in Proc. HPCN 1994, Munich, Germany April 1994. Volume 2, PP137. http://www.infomall.org

[FFHNS93] Fox. G.C., Furmanski, W., Hornberger, P., Niemiec, J., and Simoni, D., “Towards Interactive HPCC: High Performance Fortran Interpreter”, Demo at the Supercomputing’93, Portland, Oregon, 1993. http://www.npac.syr.edu/PROJECTS/PUB/wojtek/hpsin/hpfi.html

[FFHNS94] Fox. G.C., Furmanski, W., Hornberger, P., Niemiec, J., and Simoni, D., “Implementing Televirtuality”, in *Applications of Virtual Reality*, British Computer Society, Computer Graphics and Displays Group, 1994. http://www.npac.syr.edu/PROJECTS/PUB/wojtek/hpsin/doc/tvr.ps

[FMW94] Fox, G.C., Messina, P., Williams, R., editors, *Parallel Computing Works*, Morgan and Kaufman 1994. http://www.infomall.org/npac/pcw/

[FoFu95] Fox, G.C., Furmanski, W., “WebWindows: Motivation and Application to Distributed Computing”, talk and WebTools demo presented at the CRPC Annual Review, Houston, TX, March 22, 1995. http://kayak.npac.syr.edu:2005/WebTools.html

[FRSM+93] Fox, G.C., Ranka, S., Scott, M., Malony, A.D., Browne, J., Chen, M., Choudhary, A., Cheatham, T., Cuny, J., Eigenmann, R., Fahmy, A., Foster, I., Gannon, D., Haupt, T., Karr, M., Kesselman, C., Koebel, C., Li, W., Lam, M., LeBlanc, T., OpenShaw, J., Padua, D., Polychronopulous, C., Saltz, J., Sussman, A., Wigand, G., Yelick, K., “Runtime Support for High Performance Parallel Languages”, Supercomputing’93, ftp://minerva.npac.syr.edu/PCRC/supercomputing93.ps, http://aldebaran.npac.syr.edu:1955
For a recent report, see [CoHa95].

[Furm94] W. Furmanski, “MOVIE — Multitasking Object-oriented Visual Interactive Environment”, in Fox, G.C., Messina, P., Williams, R., editors, *Parallel Computing Works*, Morgan and Kaufman 1994, http://www.npac.syr/edu/PROJECTS/PUB/wojtek/hpsin/movie.html

[Java95] Java/HotJava Programming Language and Web Browser, Alpha Release April 1995, http://java.sun.com.

[LiCh91] Li. J., and Chen, M., "Compiling communication-Efficient Programs for Massively Parallel Machines," IEEE Transaction on Parallel and Distributed Systems, pp. 361-376, July 1991. http://cs-www.bu.edu:80/faculty/mcchen/publication.html

[LuCh91] Lu, L.C., M. Chen, M., "Parallelizing Loops with Indirect Array References or Pointers," 4th Workshop on Programming Languages and Compilers for Parallel Computing, Santa Clara, California, August, 1991. http://cs-www.bu.edu:80/faculty/mcchen/publication.html

[MFCM+95] Mills, K., Fox, G., Coddington, P., Mihalas, B., Podgorny, M., “The Living Textbook and the K-12 Classroom of the Future”, .http://www.npac.syr.edu/projects/ltb.

[NHL94] Nieplocha, J., Harrison, R.J., and Littlefield, R.J., “Global Arrays: A Portable ’Shared-Memory’ Programming Model for Distributed Memory Computers”, Pacific Northwest Laboratory, in Proceedings of Supercomputing’94, http://www.emsl.pnl.gov:2080

[Podg94] Podgorny, M., “Northeast Parallel Architectures Center Computational Facilities”, http://minerva.npac.syr.edu/NPAC1/PUB/infra2/infra2.html

[VRML95] Virtual Reality Modeling Language Forum: http://vrml.wired.com