The GrADS Project: Software Support for High-Level Grid Application Development*

Francine Berman, Andrew Chien, Keith Cooper, Jack Dongarra, Ian Foster, Dennis Gannon, Lennart Johnsson, Ken Kennedy, Carl Kesselman, John Mellor-Crummey, Dan Reed, Linda Torczon, and Rich Wolski

August 1, 2001

Abstract

Advances in networking technologies will soon make it possible to use the global information infrastructure in a qualitatively different way—as a *computational* as well as an information resource. As described in the recent book "The Grid: Blueprint for a New Computing Infrastructure," this "Grid" will connect the nation's computers, databases, instruments, and people in a seamless web of computing and distributed intelligence, that can be used in an on-demand fashion as a problem-solving resource in many fields of human endeavor—and, in particular, for science and engineering.

The availability of Grid resources will give rise to dramatically new classes of applications, in which computing resources are no longer localized, but distributed, heterogeneous, and dynamic; computation is increasingly sophisticated and multidisciplinary; and computation is integrated into our daily lives, and hence subject to stricter time constraints than at present. The impact of these new applications will be pervasive, ranging from new systems for scientific inquiry, through computing support for crisis management, to the use of ambient computing to enhance personal mobile computing environments.

To realize this vision, significant scientific and technical obstacles must be overcome. Principal among these is *usability*. Because the Grid will be inherently more complex than existing computer systems, programs that execute on the Grid will reflect some of this complexity. Hence, making Grid resources useful and accessible to scientists and engineers will require new software tools that embody major advances in both the theory and practice of building Grid applications.

The goal of the *Grid Application Development Software (GrADS) Project* is to simplify distributed heterogeneous computing in the same way that the World Wide Web simplified information sharing over the Internet. To that end, the project is exploring the scientific and technical problems that must be solved to make it easier for ordinary scientific users to develop, execute, and tune applications on the Grid. In this paper we describe the vision and strategies underlying the GrADS project, including the base software architecture for Grid execution and performance monitoring, strategies and tools for construction of applications from libraries of Grid-aware components, and development of innovative new science and engineering applications that can exploit these new technologies to run effectively in Grid environments.

GrADS will foster research and technology transfer programs contributing to revolutionary new ways of utilizing the global information infrastructure as a platform for computation, changing the way scientists and engineers solve their everyday problems.

1 Introduction

Imagine remote, biodegradable sensors in the ocean, monitoring temperature, biological materials, and key chemical concentrations, transmitting the measurements via wireless technology to digital libraries of oceanographic data, mining and visualizing this data directly to derive new insights, using the refined data in large scale predictive models, redeploying the sensors to refine the system as a result of the predictions,

*This material is based upon work supported by the National Science Foundation under Grant No. 9975020.

and finally, triggering nanoactuators to remove inappropriate concentrations of effluent or other non-native materials.

Imagine an earthquake engineering system that integrates "teleobservation" and "teleoperation" to enable researchers to control experimental tools—seismographs, cameras, or robots at remote sites. Combining real-time, remote access to data generated by those tools, along with video and audio feeds, large-scale computing facilities for coupled simulation, data archives, high-performance networks, and structural models, researchers will be able to improve the seismic design of buildings, bridges, utilities, and other infrastructure in the United States.

Imagine a personal digital assistant integrated into eyeglasses, powered by body heat, and capable of calling upon ambient computing, information, and network resources so that when you enter a building, your personal information space is available to you, local computing power offloads tasks such as face recognition, translation, and navigation, and you can be simultaneously monitoring your latest earthquake engineering experiment—or your stock portfolio.

These examples illustrate what we believe will be three dominant themes in 21st Century computing: computing resources are no longer localized, but distributed—and hence heterogeneous and dynamic; computation is increasingly sophisticated and multidisciplinary; and computation is integrated into our daily lives, and hence subject to stricter time constraints than at present.

None of these examples is that far-fetched: revolutionary changes in broadband communications and wireless networking, as well as relentless miniaturization, provide the necessary technical underpinnings. Furthermore, ambitious research programs in ubiquitous computing and Grid middleware are targeting key challenges at the infrastructure level: security, resource discovery, resource management, power management, and the like. However, existing efforts are not addressing one fundamental problem: the programming of these highly complex and dynamic systems. This challenging problem is the focus of the *Grid Application Development Software* (GrADS) project, established by the authors with support from the NSF Next Generation Software Program in 1999. In this paper, we present the vision and strategies underlying the GrADS effort.

The Grid Programming Problem

Our use of the term "Grid" is inspired by a recently-published volume entitled "The Grid: Blueprint for a New Computing Infrastructure" [24], which established a compelling vision of a computational and information resource that will change the way that everyone, from scientist and engineer to business professional, teacher, and citizen uses computation [63, 24]. Just as the Internet defines fundamental protocols that ensure uniform and quasi-ubiquitous access to communication, so the Grid will provide uniform access to computation, data, sensors, and other resources. Grid concepts are being pursued aggressively by many groups and are at the heart of major application projects and infrastructure deployment efforts, such as NASA's Information Power Grid (IPG) [40], the NSF PACI's National Technology Grid [63] and Distributed Terascale Facility, the NSF's

Grid Physics Network, and the European Union's EU Data Grid and Eurogrid projects. *These and many other groups recognize the tremendous potential of an infrastructure that allows us to conjoin disparate and powerful resources dynamically to meet user needs.*

Despite this tremendous potential, enthusiasm, and commitment to the Grid paradigm, the dynamic and complex nature of the Grid environment, and the sophistication of the applications being discussed, the challenges are daunting. Few software tools exist. Our understanding of algorithms and methods is extremely limited. Middleware exists, but its suitability for a broad class of applications remains unconfirmed. Impressive applications have been developed, but only by teams of specialists [48, 15, 23, 24, 28, 44, 59]. Entirely new approaches to software development and programming are required for Grid computing to become broadly accessible.

It is this combination of the national importance of the problem and the need for multidisciplinary research advances that has led the authors to initiate the GrADS project and focus it on the goal of conducting fundamental research leading to the development, in prototype form, of technologies needed to make the Grid usable on a daily basis by scientists and engineers. Collectively, the challenges that must be overcome to achieve this goal can be summarized in a single requirement: We need application development technologies that make it easy to construct and execute applications with reliable [and often high] performance in the constantly-changing environment of the Grid.

As we pursue this goal, we can draw upon a significant body of knowledge and technology in distributed computing, the Internet, and Grid middleware. However, while traditional distributed computing technologies provide critical building blocks and frameworks for Grid application development, distributed computing is not concerned with, and does not address the large-scale and dynamic resource sharing, frequently stringent performance requirements, large resource needs, and the multidisciplinary nature of Grid applications. Although emerging Internet, peer-to-peer, and Grid middleware technologies are meeting the need for large-scale resource sharing, they do nothing to simplify application development.

The GrADS project has begun to develop the knowledge and technology base required to support application execution in this new computing environment, along with application development strategies to make it accessible to ordinary scientists, engineers, and software developers for problem solving. To do this, we are pursuing research in four major areas: (1) collaboration on the design and implementation in prototype form of important scientific applications for the Grid; (2) the design of programming systems and problem-solving environments that support the development of configurable Grid applications by end users in high-level languages close to the notation of their application domain; (3) the design and implementation of execution environments that dynamically match configurable applications to available resources in order to provide consistent, reliable performance; and (4) the design and construction of hardware and software testbeds for experimentation with the GrADS program preparation and execution system and the applications developed to use them.

We anticipate that the successful completion of this research program will lead to revolutionary new ways of utilizing the global information infrastructure as a platform for computation, data sharing, and collaboration.

1.1 Applications

Just as the emergence and usability of the World Wide Web has ushered in new paradigms in application development and access to information, the maturing of the Grid and its natural extension to peer-to-peer platforms, wireless endpoints, remote instruments, and sensors will engender innovative new application paradigms and new environments for application development and execution. Such environments will support application adaptivity, portability, ubiquity and performance. Emerging Grid applications will provide the driving force behind the architecture, research and prototypes that will be developed by researchers in the Center for Grid Application Development Software. Over the next decade, Grid applications will address a wide variety of critical challenges in science and engineering. New applications in computational biology, bioinformatics, genomics, high energy physics, crisis management, and other domains will require real-time data collection, mining and analysis, simulation, and visualization of results.

There are several key challenges that must be addressed for Grid computing to be effective. Applications must be able to nimbly adapt to a dynamic set of target resources and to incorporate huge amounts of information from heterogeneous sources and distant endpoints (sensors, target computational resources on peer-to-peer networks). Moreover, the Grid software infrastructure to which the applications themselves are targeted is complex, heterogeneous, and dynamic. Globus [25], NetSolve [8], Condor [45, 46], Legion [33, 30], and commercial infrastructure systems have different levels of robustness and operate on different resource subsets. Over the next ten years, applications will need to be able to adapt and perform with respect to the infrastructure provided by ambient resources. The design of development environments and run-time systems for such adaptable and "ultra-portable" applications constitutes an extremely challenging and comprehensive set of problems.

The success of the Grid as a computing platform is dependent on development of performance-efficient applications that can effectively exploit a wide range of cooperating resources. Software that supports development and execution of such applications is critical to making Grid programming tractable. During the research associated with the Center for Grid Application Development Software, a collection of emerging Grid-enabled applications will focus our research goals, and help set research priorities. These applications provide a means for critical evaluation and assessment of the Grid application development software resulting from our research. The following examples are representative of major classes of a new generation of Grid applications.

On-Demand Applications A critical aspect of computational Grids is their ability to concentrate the massive computational and information resources required for real-time, on-demand applications. To understand the importance of on-demand application development for the Grid, consider the problem faced by a crisis manager after a major disaster such as an earthquake. Although the component operations that are essential to crisis management are known in advance, each crisis presents unique requirements. The crisis manager must integrate information from many different sources to determine the actions needed to respond to the particular crisis at hand. For example, she must be able to understand the state of the current infrastructure, possibly by aggregating sensors in buildings, power lines, and utility conduits into a network that can report the changing state of the basic infrastructure. She must be able to integrate reports from emergency crews with patient information to ensure that emergency treatments are consistent with the needs of each patient. Finally, she needs to be able to access mesoscale weather models, fueled by information from a grid of Doppler radars, to identify weather patterns that may exacerbate the crisis. There may also be a need to simulate the flow of groundwater contaminants through the soil.

Ubiquitous Applications During the next decade, an increasing number of users will develop applications for execution on a platform where the user does not know (or care) where the application might be executed. Current examples of such software platforms include SETI@home [22] and Entropia [19] (which target largely embarrassingly parallel applications to compute on "throw away" endpoints), Condor (which targets individual, migratable and largely embarrassingly parallel jobs on workstation clusters) and APST (middleware that targets parameter sweep applications on a wide variety of grid environments). Such systems demonstrate the potential for the Grid, but the research community must improve application programming models for Grid execution. As part of the GrADS research, we intend to develop more sophisticated (and dependable) programming models that can execute ubiquitously and reliably in large-scale Grid environments.

Robust, Portable Applications Much as the World-Wide Web has catalyzed the creation of immense collections of private data, and supports easy accessibility to large quantities of public data, we anticipate that the emergence of the Grid will spawn public Grid resources (computing, storage, etc.). Already, we see significant development underway for production Grid systems (e.g., the NASA IPG [32], NSF GriPhyN [34] and NEESgrid [21], and European Data Grid [31]) based on a existing software infrastructures (e.g., Globus in the four examples just cited). To capitalize on such resources, a new generation of portable, Grid-aware applications is needed.

The user community's desire to exploit Grid resources and to protect their software development investment is important driver for developing portable, standard services that support robust Grid applications. In the long run, convincing the developers of Grid applications—users, scientists, and third party commercial organizations (ISVs)—must depend on the development of standard interfaces for critical Grid services that enable both *portability* forward to new software infrastructures and platforms and *access* to very large numbers of resources. However, current Grid software infrastructures lack the capabilities to support flexible, robust Grid applications in a world of heterogeneous systems, unreliable networks, and asynchronous resource revocation. The core research that has begun under the aegis of the GrADS project includes support for adaptivity, resource negotiation, and performance contracts. These capabilities will help applications operate effectively in an ever-changing Grid environment. In addition, the proposed research will develop the understanding that enables definition of standard shared libraries that export these capabilities.

Integrated Data Analysis and Simulation Data-oriented applications will constitute one of the most active and critical areas for the next decade in science and engineering. Many research communities collect, analyze, and mine immense amounts of data in collections that are often not co-located with the computational servers. For example, there is considerable effort currently being devoted to the development of parallel and distributed applications that use genomic data to assess, evaluate and develop structural models and to answer fundamental questions about life.

In addition, the GriPhyN [34, 11] project is developing a distributed analysis environment for physics experiments that will serve thousands of users. A crucial concept being pioneered by GriPhyN is virtual data, i.e., derived data products that are defined by the computations to produce them. Given a set of virtual data definitions, a user request for data value(s) can be translated into computations and data movements. We anticipate collaboration with GriPhyN in two areas: estimation of the computational requirements of virtual data computations, along with scheduling of computations and data movement based on compiler-detected query profiles.

Another important area in which integrated data assimilation from distributed resources is becoming more important is in weather forecasting, exemplified by the Integrated Forecasting System (IFS) developed by European Center for Medium Range Weather Forecasting (ECMWF), and in climate analysis, exemplified by the ERA-40 project covering the time period from 1957 - 2001 pursued jointly between NCAR, NOAA, NESDIS, ECMWF and several other organizations. The IFS has real-time aspects and uses a wide range of sensor and network technologies and is an excellent application for GrADSoft technologies.

The GrADS project is collaborating with both developing applications and mature Grid exemplar codes to guide our design and development efforts. In the long term we plan to work with developers of exemplar applications in each of the application classes to prototype program development software that meets the needs of current Grid applications as well as the new generation of applications that evolve to reap the benefits of the Grid. During our research into application development software, we expect to gain new insights into how to design and implement grid applications. Thus, our research agenda includes the study of new types of applications, as well as new approaches to application design and implementation.

1.2 Vision

For the Grid to become a really useful computational environment—one that will be routinely employed by ordinary scientists, engineers, and other problem solvers—it must be relatively easy to develop new applications. Currently applications must be developed atop existing software infrastructures, such as Globus, by developers who are experts on Grid software implementation. Although many useful applications have been produced this way, it is too difficult for Grid computing to achieve widespread acceptance.

In our vision, the end user should be able to specify applications in high-level, domain-specific problem-solving languages and expect these applications to seamlessly access the Grid to find required resources when needed. In these environments, users would be free to concentrate on how to solve a problem rather than on how to map a solution onto the available Grid resources.

To realize this vision we must solve two fundamental technical problems. First, we must understand how to build programming interfaces that insulate the end user from the underlying complexity of the Grid execution environment without sacrificing application execution efficiency. Second, we must provide an execution environment that automatically adapts the application to the dynamically-changing resources of the Grid. Our overall approach to addressing these challenges is described in the next section.

2 The GrADS Software Architecture

To address the fundamental challenge of program development for Grid environments, GrADS has initiated a coordinated and far-reaching program of research, prototyping, and technology transfer aimed at the central problems of programming models, algorithms, programming systems, and applications.

Underlying and unifying our diverse investigations is a basic assumption: that effective application development for Grid environments requires a new approach to the design, implementation, execution, and optimization of applications. A new strategy is needed because the traditional development cycle of separate code, compile, link, and execute stages assumes that the properties of underlying resources are static and relatively simple. In the Grid, this assumption is not valid. (Needless to say, the alternative approach, frequently adopted in distributed computing, of hand coding applications with socket calls or remote procedure calls is not viable either.) We require a software development environment that enables the effects of dynamism to be mitigated and controlled.

Figure 1 presents the new program development structure that we believe is required. In what we refer to as the GrADSoft architecture, the discrete steps of application creation, compilation, execution, and post-mortem analysis are replaced with a continuous process of adapting applications to a changing Grid and to a specific problem instance. Two key concepts are critical to the working of this system. First, an application must be encapsulated as a *configurable object program*, which can be optimized rapidly for execution on a specific collection of Grid resources. Second, the system relies upon *performance contracts* that specify the expected performance of modules as a function of available resources. Our research and development effort has begun to address the various elements of this architecture. In the remainder of the paper, we summarize the key ideas underlying the GrADS effort and explain in detail the technical challenges to be addressed and the approach to be followed in each area.

GrADS Program Preparation System. The left side of Figure 1 depicts the tools used to construct configurable object programs. We expect that most application developers will use high-level problem solving environments (PSEs) to assemble Grid applications from a toolkit of domain-specific components. Another path allows developers to build the specialized components that form these PSE toolkits (*e.g.,* a library for solving PDEs on computational grids) or to create new modules for their specific problem domain.

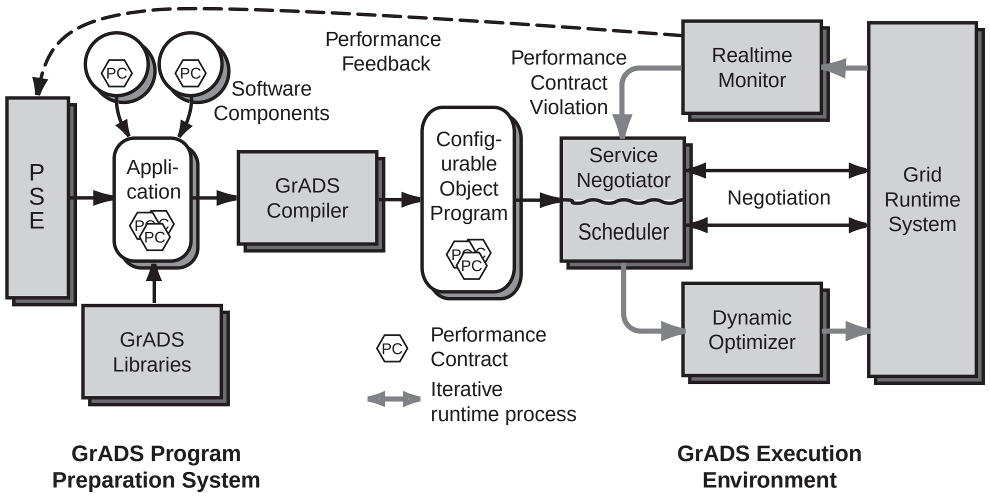

Figure 1: GrADS Program Preparation and Execution Architecture

In either scenario, modules are written in derivatives of standard languages with Grid-specific extensions (*e.g.,* data or task distribution primitives). They are bound together into larger components, libraries, and applications with a coordination language. This process creates malleable modules, annotated with information about their resource needs and predicted performance for a wide variety of resource configurations.

The goal is to build tools that free the user from many of the low-level concerns that arise in programming for the Grid today, and to permit the user to focus on high-level design and performance tuning for the heterogeneous distributed computing environment.

GrADS Execution Environment. When a configurable object program is delivered to the execution environment, the GrADSoft infrastructure must first determine what resources are available and secure an appropriate subset for the application. Using annotations from performance contracts and results from compiler analysis, service negotiators will broker the allocation and scheduling of module components on Grid resources. Next, the infrastructure will invoke the dynamic optimizer to tailor the reconfigurable object program for good performance with the available resources. This step will also insert sensors and actuators to help the performance monitoring system control application execution.

During program execution, a real-time monitor tracks program behavior and validates observed behavior against performance contract guarantees. Should a performance contract be violated, the monitor will respond by interrupting execution through an actuator, leading to several possible actions. The actuator can invoke the dynamic optimizer with more information (from performance monitoring) to improve program behavior in the current execution context, negotiate a new execution context where the existing executable is more likely to satisfy the old contract, or do both by negotiating a new context and tailoring the executable for it. Dynamic forecasts of resource performance and Grid capacity will be used to reduce renegotiation overhead. The goal of this closed loop system is to ensure that execution of the application proceeds reliably, meeting the specifications of its performance contracts, in the constantly changing Grid environment.

3 Program Preparation System

Developing a parallel program for efficient execution on the Grid currently requires a level of expertise that few possess. Unless Grid programming can be greatly simplified, the power of Grid computing will be inaccessible to many. To address this issue, the GrADS project is conducting research on program preparation systems, focusing on the design and construction of software that simplifies building and running Grid-enabled applications.

To simplify development of Grid-enabled applications, we are focusing on a methodology in which most users will compose applications in a high-level, domain-specific language built upon pre-existing component libraries. This approach hides Grid-level details and lets the user express a computation in terms that make sense to an expert in the application domain. Underneath the domain-specific language, and supporting it, will be a layer of software that manages the complex task of computing on the Grid. This software, embedded in a collection of libraries, will include not only the base algorithms, but also composable performance models and dynamic mapping strategies for each method. To manage heterogeneity and the late binding of resources without sacrificing performance, we are developing a dynamic optimizer that does load-time code optimization.

For this approach to succeed, we are undertaking the following scientific and technical challenges: (1) development of programming models and compiler technology to support efficient high-level programming; (2) development of programming models that help library writers cope with properties of the Grid such as variation in latency and performance, or even failure; (3) design of composable performance models for use in selecting resources, in tailoring code to runtime resources and in detecting performance problems; (4) incorporation of partial evaluation strategies in the GrADS compiler to support rapid runtime tailoring for efficient execution; and (5) investigation of the impact of essential activities such as checkpointing, reporting, and monitoring on overall performance and devising strategies to mitigate these effects. These issues cannot be addressed at a chalkboard. To find appropriate solutions we have adopted a methodology that includes extensive experimentation and exploration in the context of the GrADS applications effort. By repeatedly moving solutions into prototype tools and using these tools in the next generation of applications, we will refine our approaches and improve both their effectiveness and usability.

A Framework for Grid Application Development A Grid programming system should make it easy for end users to build applications that execute efficiently on the Grid. Such a systems should provide several ways to construct applications. We expect that the most common approach will be to compose applications from pre-written, domain-specific, library components as is done with CCAT [29], Khoros [43], SciRun [7] and NetSolve [7]. These systems allow users to “script” an application by configuring and composing software components or services that run elsewhere into a single distributed application. Scripts are either composed graphically, or they are written using a high level scripting language like Python [47] or Matlab [37].

To bridge the gap between these comfortable, high-level, scripting languages and an efficient, Grid-enabled

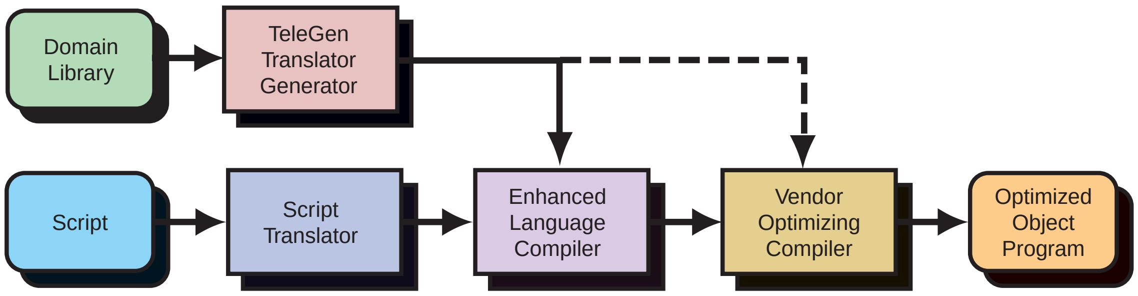
Domain Library
TeleGen Translator Generator
Script
Script Translator
Enhanced Language Compiler
Vendor Optimizing Compiler
Optimized Object Program

Figure 2: Telescoping Languages

executable, we must develop a novel and effective compilation system. Our strategy has two novel components: an implementation technique for the domain-specific languages that we call *telescoping languages* [42], and a tool for load-time tailoring that we call the *dynamic optimizer*.

Telescoping Languages The telescoping languages approach makes extensive use of whole-program analysis and optimization to automate the construction of extensible languages. We will build a system called the *TeleGen* compiler, shown in Figure 2. TeleGen will read an annotated, domain-specific library, analyze it, and produce a customized optimizer that understands the library entry points (including their execution properties) as if they were native primitives in the base language. It will also create a version of the library that includes optimized versions of some entries. TeleGen's compilation approach builds upon work in high-level axiom-driven optimization [50, 49], call-site analysis and library routine implementation selection [36], and interprocedural optimization of high-level languages [16, 10].

The resulting optimizer behaves as a compiler for a new language—the base language (*e.g.,* C) augmented with the functions of the domain-specific language. In this scheme, a domain-specific scripting language can be implemented as a preprocessor that translates scripts into base language programs that call the library components. The optimizer should produce highly optimized code from such an input. This implementation strategy can be applied iteratively to several different levels of libraries—telescoping them into one translator.

The success of the telescoping languages strategy depends upon the existence of sophisticated component libraries that are specially prepared for Grid execution by professional library developers. (The challenges of developing such libraries are detailed below.) These libraries will be annotated by the developers with mapping strategies, performance models, hints on how to optimize calls to individual components in various contexts, and algebraic specifications of library operations (*e.g.*, commutativity, transitivity, associativity [69]) to facilitate high-level optimization.

A critical issue for this work is constructing TeleGen so that the optimizers it generates can, themselves, produce configurable object programs suitable for use in the GrADS execution system. The optimizer must understand all of the components of a library—code, performance model, and mapping strategy—and must manipulate them to create the configurable object program.

Research issues include the design of high-level optimizers for the Grid, methods for selecting the right code variants for a given collection of Grid resources, mechanisms for generating and managing the myriad variants that the system will need, and the design of tools to help the library designer build useful library annotations.

Dynamic Optimizer The dynamic optimizer is a component of the program preparation system that lives in the execution environment. It is invoked at load time to tailor the configurable object program to the actual runtime environment. The dynamic optimizer queries the target machine for configuration data, inserts the sensors and actuators needed by the runtime system, and rewrites the object program into an executable that will run efficiently on the target machine. Deferring code generation into the dynamic optimizer, should simplify configurable object programs, reduce their size, and provide more consistent optimization.

Libraries and Algorithms Modeling, simulation, and data intensive computing have become staples of scientific research. This has exposed the difficult aspects of scientific computing to a broader audience of scientists and engineers. While access to computing has improved dramatically over the past decade, efficient scientific computing still requires specialized knowledge in numerical analysis, computer architectures, and programming languages. Many working researchers do not have the time, the energy, or the inclination to acquire such expertise. Scientists expect their computing tools to serve them, not the other way around. Unfortunately, the growing desire to tackle interdisciplinary problems with more realistic simulations on increasingly complex computing platforms, will only exacerbate the problem. The classic solution to this problem was to encode the requisite expertise into easily used libraries. While traditional numerical libraries (e.g. LAPACK [4], Ellpack [38], ScaLAPACK [6], PETSc [5]) have brought immense benefits, the radical changes occurring in scientific computing are creating new challenges that these libraries, in their current form, cannot meet. that despite the immense benefits such traditional numerical libraries

To address this challenge, we are developing a new generation of *Self-adapting Numerical Software (SaNS)* systems. These SaNS will not only meet today's challenges; they should address future changes in scientific computing as well. We will design and build a framework for SaNS for numerical libraries and algorithms. This system will operate as "black box" software; intended for use by domain scientists who need not understand the algorithmic and programming complexities it encapsulates. To manage the complexities of the Grid and to adapt in ways that maximizes their effectiveness, SaNS must encapsulate far more intelligence than standard libraries. The work described below will make it possible to produce a SaNS system that: (1) automatically analyzes the logical and numerical structure of the data to let the library choose the best algorithmic strategy for solving the problem; (2) embodies a set of rules, progressively self-tuned over time, for choosing the appropriate algorithm for a given linear system, based on its analysis of the data and any hints provided by the user; (3) encodes metadata about the user's data, about its own characteristics, and about the known implementations of the algorithm it selects, so that the system can schedule the computation effectively on the available resources; and (4) uses a scripting language that generalizes the decision procedure that the SaNS follows and enable scientific programmers to easily make use of it.

Using SaNS libraries should improve the ability of computational scientists to solve challenging problems efficiently—without requiring much extra-domain expertise. As these innovations become generally available, they will create a dynamic computational environment that automatically selects and integrates the most effective library components for a given problem, data set, and collection of resources. The SaNS metadata scheme will let us capture this self-adaptive process in databases, creating an indispensable resource for future library developers. Current numerical libraries, whose limitations are increasingly obvious, are now threatened with obsolescence. This investigation will lay the foundation needed to meet the challenging demands of computational science over the next decade.

4 An Execution Environment for Grid Applications

As we explained above, our research seeks to allow future Grid applications to operate in highly dynamic environments, adapting their resource demands and behavior to the environments in which they find themselves—and also, when possible, adapting the environment to fit their requirements.

The realization of this overall goal requires the development of new mechanisms for information and control flow between program preparation system, program, and environment, so that (1) information about the environment, and program behavior in that environment, can be discovered and communicated to program components in meaningful terms, and (2) program requirements can be communicated to the environment, and to program components, in ways that admits to effective control.

These two goals define, collectively, the purpose of the GrADSoft execution environment. They have led us to focus our research in this area on three key issues, namely the protocols, services, and methods required to (1) discover and disseminate information about the dynamically changing structure and state of Grid resources (*Grid information service*); (2) select, allocate, and control collections of Grid resources, and communicate requirements among resource providers and consumers (*resource management service*); and (3) monitor and, as necessary, control adaptively an executing program. These protocols and services represent what is sometimes called middleware [1]: code that executes in the network in support of applications. As in other areas of the Grid, we are concerned with achieving a separation of concerns between *resource protocols* that must be broadly deployed and *collective protocols and services* that can be localized in more application-specific code [27].

The following scenario illustrates some of the issues that arise in the Execution Environment. We imagine the Program Preparation System generating an executable image, a set of performance requirements, and a budget for executing the application that is expressed in some Grid currency. These latter two abstractions serve as the basis of a "performance contract" between the application and the resources it uses. The Execution Environment then launches the program by submitting it to the *application monitoring and adaptive control service* (service (3)). To do so, this system consults the *Grid information service* (service (1)) to determine what resources are available and appropriate, and the *resource management service* (service

(2)) to ensure that those resources are allocated for the execution, subject to demand and supply respectively.

In pursuing these goals, we are working initially within the context of the Grid architecture defined by the Globus Toolkit [25], due its widespread adoption within the scientific community, and significant experience and code base. At the Connectivity and Resource levels in a Grid architecture, the Globus Toolkit defines standard authentication and authorization protocols, information service protocols, and resource management protocols. At the Collective level, it provides resource discovery, brokering, and co-allocation functions. (Other relevant protocols and services are being discussed within the Global Grid Forum, for example for event delivery.) The adoption of this framework has allowed us to focus our attention on the central problems (for us) of how to obtain, organize, and exploit monitoring and control information, problems that can be expressed in terms of interactions among cooperating services and resources. Issues of security, resource access, and the like can be relegated to Globus—and/or to complementary industrial standards and trends such as Jini [67] and the emerging peer-to-peer technology base (as is being pioneered by companies such as Entropia, CDDB, Napster, and Parabon [19, 9, 51, 53]). The end result of this work will be the definition of both a middleware architecture and specific new middleware services designed to support the concerns of adaptive Grid computations. We expect that this work will result in useful feedback to the Grid protocols and services R&D community.

In the following, we expand upon each of the three points noted above, indicating in each case the nature of the primary research challenges.

Grid Information Service To provide the functionality needed for negotiation and scheduling, the Execution Environment must be able to obtain information about the resources available for application use. A wide variety of information can conceivably be of interest: for example, hardware configuration, measured load, access control policies, application performance data, power consumption [52], estimates of nonobservable system properties, and predictions of future states [70, 17, 61, 18, 52, 41]. Our goals in the GrADS project is to first to develop an integrated framework in which these many different types of information can be used in a coordinated and uniform fashion, second to conduct a broad exploration of how different sorts of information can be produced and used, and hence, third, to produce a set of effective techniques for information collection, analysis, and application.

In previous work, we have established frameworks for providing uniform access to, and indexing of, diverse information sources (the Globus MDS [20, 12]), for collecting experimental data and using this data to generate forecasts of future state (the Network Weather Service: NWS [70]), and for structuring networks of sensors and transformers to support adaptive control (Autopilot [56]). Each of these systems has been proven effective in various experimental and (in some cases, e.g., MDS), large-scale deployments. Within GrADS, we are building on this infrastructure, integrating these diverse elements and extending them in major ways. We are developing new services, including distributed event management, new methods of measuring and predicting components of system state, methods for discovering and maintaining relevant information about resources of interest in the execution environment, robust and scalable publication methods, methods that can deal effectively with both measured and dynamically derived data, and methods for information service discovery in widely distributed, dynamic environments [65, 14, 35, 39]. We are also addressing the question of how to represent our degree of confidence in data and security concerns relating to dissemination of data.

Grid Resource Management Service The Execution Environment must also provide the ability to reserve, allocate, configure, and manage collections of resources that match an application's needs. Building on elements of the Globus resource management architecture [13], which provides secure remote access and reservation [26] mechanisms, we are developing new co-reservation and co-allocation algorithms capable of dealing with resources with dynamic and probabilistic properties, integrate performance contracts (see next paragraph) into resource reservation and resource operations, integrate traditional quality of service methods into resource management frameworks, and map compiler-derived and library-derived performance information into global resource reservation and allocation services.

A major goal of our work in this area is to explore and understand the nature of the language that should be used to share complex, multi-dimensional requirements and performance data among resource providers and consumers. To that end, we are investigating the design of a language of *performance contracts* to enable dynamic negotiation among resource providers and consumers. A performance contract maps a set of resources and a set of application resource needs to a specified performance level—to satisfy the contract, the assigned resources and the application must behave as specified.

Our approach to performance contracts derives them from a performance model provided by the configurable object program and a set of resource performance characteristics culled from the Grid information service. The *service negotiator* (which is logically part of the *application monitoring and adaptive control service*) brokers performance contracts between applications and resources. It uses the information and reservation services to find available resources, select a set that matches the predicted needs of the application, and make any needed reservations. As a part of our research, we have begun to develop a theory of performance contracts and service negotiation that can be adapted to the varying behavior of the Grid. Matchmaking techniques may be relevant here [45]; see also [64].

Building on this framework, we will investigate more dynamic resource brokering mechanisms based on the use of economic models (*e.g.,* bidding, cost negotiation, and dynamic pricing) as a basis for arbitrating between competing resource demands. We plan to study both auction-based and commodity-based formulations of the performance economies. Auction-based systems are attractive because of their scalability, but it can be shown that commodity-based (but not auction-based) economies achieve both equilibrium and stability [68, 60]. Since Grid applications must adapt to changing performance conditions, overall system stability is an important concern.

Application Monitoring and Adaptive Control Service Work in the two areas just listed will provide a powerful, extensible framework for communicating requirements, various information, and control functions among applications, intermediate brokering functions, and resources. The third area in which we are working is building on this framework to construct a a closed-loop control system, called the *application execution monitor* that uses various dynamic performance information sources to guide an application to completion despite performance variations in the underlying resource base, via a process of adaptive control of both application behavior and resource demands.

To enable such adaptation, the execution monitor depends on the *dynamic optimizer* (which will be developed in conjunction with the Program Preparation System) to insert the sensors and actuators that let it manage the execution. The dynamic optimizer, invoked just prior to execution, also instantiates the final performance contract according to the rules of the resource economy that is in place.

The Autopilot system [57, 55, 56] embodies several of the ideas on which we will build our distributed monitoring systems. Autopilot sensors, inserted in application or library code, can capture application or system characterization metrics. When an application executes, the embedded sensors register with a directory service provided by an Autopilot Manager. Sensor clients can then query the manager to locate sensors with specific properties and receive measurement information directly from these sensors. Sensors, managers and sensor clients can execute anywhere on the Grid.

Atop this substrate, the key research issue is developing techniques to decide how and when a performance contract has been violated (*e.g.,* managing temporal variation and distributed contract testing) and how to respond to the violation in order to maximize application performance. To carry out this plan, we are investigating new strategies that let the compiler, scheduler, runtime system, and other components cooperate to extract, non-intrusively, pertinent information from the running application.

Our preliminary experiments with performance contracts [66] indicate that this is a fruitful approach. Using Autopilot's fuzzy logic decision procedures, it was possible to detect local perturbations in processor and network availabilities during application execution. The current focus of our work is to extend these tests to encompass the temporal and global contract aspects that we mentioned above.

5 Understanding Grid Software Behavior

The long-term success of our Grid software research agenda requires that we develop design methodologies that allow systematic design and evaluation of dependable, robust, and scalable Grid services and applications software. Unfortunately, such design methodologies are currently totally lacking. It is no exaggeration to say that Grid services and software are designed and characterized today largely based on the designer's intuition and on ad hoc experimentation with little knowledge of *when* they will fail catastrophically. We view this as completely unsatisfactory and adopt as our long-term research goal the development of an experimental methodology for characterizing grid software that allows us to evaluate and predict the performance, fault tolerance, and scalability of middleware services.

As an important first step towards the development of such design methodologies, we are developing and deploying two major testbeds and associated tool suites designed to provide both soft (configurable) and hard (fixed) environments for exploring dynamic Grid behaviors. Our goal in this work is to enable systematic study and ultimately understanding of the dynamic behavior of Grid resources, middleware, and applications.

These tool testbeds and tool suites are referred to as the *MicroGrid* and *MacroGrid* testbeds. Both share the use of Globus services as a unifying computational environment. They differ in terms of the degree of configurability and realism they offer. The use of a common environment means that programs can be run without change on both testbeds, hence allowing comparative studies.

The MicroGrid testbed, which runs on clusters of PCs or workstations, provides tools that use a combination of simulation and direct execution techniques to produce a repeatable, observable testbed for Grid experiments. Major challenges here include:

- *Fidelity in Grid Resource Modeling* The modeling of computation, storage, and networking resources faithfully, across a range of resource requirements and execution speeds. Scalable online network simulation is a critical challenge—and differs from the offline simulation efforts generally studied by the networking research community.

- *Representative Background Load Modeling* Understanding what interaction of background and foreground load is critical to representative behavior. This is essential to all aspects of resource modeling, including computation, storage, and network systems.

- *Efficiency and Scalability* Achieving efficient simulation to enable study of long periods of behavior, and scalability to achieve the study of large systems—which often exhibit different behavior.

As part of the GrADS effort, we have constructed and are experimenting with a number of generations of the MicroGrid tools [62], exhibiting a succession of greater capabilities. These efforts are integrating our novel research efforts with relevant efforts developed in the community.

The second major infrastructure, the MacroGrid, integrates computational and network resources at the GrADS institutions to serve as a realistic (although less configurable) experimental testbed. This testbed provides a more controlled environment, and likely a much higher degree of instrumentation and data capture, than is possible in typical Grid environments. This testbed is being used for the initial applications experiments discussed in the next section and to validate MicroGrid simulations.

Future efforts will focus on developing an experimental methodology for characterizing grid software in a manner that allows accurate evaluation of the software's behavior before deployment. A further goal of this work is understanding how to characterize a regime of behavior and also to identify those regimes for which behavior is poor, or at least uncharacterized. Possible approaches include statistical sampling, perturbation analysis, and enforcement of behavioral constraints (e.g., linearity) on software.

6 Research Methodology and Progress

The GrADS research and development activities are organized as three parallel, interdependent thrusts: basic research, testbed development, and application evaluation. The basic research thrust began by defining performance contracts, exploring adaptivity (both experimentally and theoretically), and creating initial prototypes of the GrADS development and execution environment. A key step in this effort was the investigation of two application prototypes discussed in other papers within this volume: a distributed version of the ScaLAPACK linear system solver [54] and a Grid-enabled version of Cactus [2, 3, 58], a powerful modular toolkit for the construction of parallel solvers for partial differential equations. With knowledge gleaned from these efforts, the research thrust has begun to explore an integrated approach to Grid software development that emphasizes compile-time and runtime information sharing among algorithms, compilers, tools, and libraries.

The testbed development thrust has embarked on the creation of a substantive GrADS software toolkit (GrADSoft) that provides a basis for experimental verification of our ideas and for technology transfer. Exploiting the core infrastructure provided by Globus and software components from our AppLeS, NWS, Autopilot, NetSolve, D95, and scalar compiler systems, the GrADSoft prototype will eventually bring together an increasingly sophisticated set of languages, libraries, compilers, schedulers, service negotiators, and performance tools.

Finally, in concert with our PACI, ASCI, and other partners, the evaluation thrust will use the emerging GrADSoft prototype to develop and assess Grid-enabled applications. This evaluation will couple the basic research and testbed efforts and provide a blueprint for a powerful technology transfer mechanism for the GrADS project and possible extensions thereof. The Cactus experiment, alluded to above, is an example of this approach.

In brief, the research, testbed, and application evaluation thrusts are linked a tight cycle of exploration, development, and experimental validation that focuses research on problems that are both important and practical.

7 Summary

The GrADS project has established an effort to pioneer technologies that will be needed for ordinary scientific users to develop applications for the Grid. These technologies will include a new program preparation framework and an execution environment that employs continuous monitoring to ensure that reasonable progress is being made toward completion of a computation.

Based on preliminary efforts to develop Grid-enabled versions of the ScaLAPACK linear solver and the Cactus toolkit, we have begun construction of the GrADSoft infrastructure, which will provide generic mechanisms for initiating and monitoring the execution of applications on the Grid. In addition, the project has developed and defined the concept of a "configurable object program" which includes the resource mapping and performance modeling components necessary for use in the GrADS execution system. Finally we have constructed two major testbeds, the MicroGrid and the MacroGrid, to support experimentation with Grid execution and monitoring technologies.

In the future, we plan to address the programmability problem through the development of frameworks for generating high-level, domain-specific problem-solving systems based on libraries of Grid-aware components. These libraries are the subject of a major research thrust of the GrADS effort.

Over the long term, we believe that a system like the one being constructed by GrADS can dramatically increase the impact of the Grid by making it accessible to the entire science and engineering community.

Acknowledgments

The authors would like to thank Frederica Darema, manager of the NSF Next Generation Software program, which provides GrADS funding, for her constant support of our effort and her intellectual contributions to the ideas underlying the GrADS project. In addition, we would like to thank the the many participants (research staff and students) in the GrADS project who have contributed to the development of the ideas and systems described in this paper.

References

[1] R. Aiken, M. Carey, B. Carpenter, I. Foster, C. Lynch, J. Mambretti, R. Moore, J. Strasnner, and B. Teitelbaum. Network Policy and Services: A Report of a Workshop on Middleware. IETF, RFC 2768, http://www.ietf.org/rfc/rfc2768.txt, 2000.

[2] G. Allen, W. Benger, T. Goodale, H. Hege, G. Lanfermann, A. Merzky, T. Radke, and E. Seidel. The Cactus code: A problem solving environment for the grid, 2000.

[3] Gabrielle Allen, David Angulo, Ian Foster, Gerd Lanfermann, Chuang Liu, Thomas Radke, Ed Seidel, and John Shalf. The cactus worm: Experiments with dynamic resource discovery and allocation in a grid environment. *International Journal of Supercomputer Applications*, 2001.

[4] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *"LAPACK Users' Guide, Third Edition"*. SIAM, Philadelphia, PA, 1999.

[5] Satish Balay, William D. Gropp, Lois Curfman McInnes, and Barry F. Smith. PETSc users manual. Technical Report ANL-95/11 - Revision 2.1.0, Argonne National Laboratory, 2001.

[6] L. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. Whaley. *ScaLAPACK Users' Guide*. SIAM, Philadelphia, PA, 1997.

[7] H. Casanova, J. Dongarra, C. Johnson, and M. Miller. Application–Specific Tools. In I. Foster and C. Kesselman, editors, *The Grid: Blueprint for a New Computing Infrastructure*, pages 159–180. Morgan Kaufmann, 1998.

[8] Henri Casanova and Jack Dongarra. NetSolve: A network-enabled server for solving computational science problems. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(3):212–223, Fall 1997.

[9] CDDB, `http://www.cddb.com`.

[10] S. Chauveau and F. Bodin. Menhir: An environment for high performance Matlab. *Scientific Programming*, 7:303–312, 1999.

[11] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke. The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Data Sets. *Journal of Network and Computer Applications*, 2000.

[12] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid information services for distributed resource sharing. In *Proceedings of the 10th IEEE Symposium on High-Performance Distributed Computing (HPDC)*. IEEE Computer Society Press, 2001.

[13] K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke. A Resource Management Architecture for Metacomputing Systems. In *Proceedings of the Fourth Workshop on Job Scheduling Strategies for Parallel Processing*, 1998.

[14] S. E. Czerwinski, B. Y. Zhao, T. D. Hodes, A. D. Joseph, and R. H. Katz. An Architecture for a Secure Service Discovery Service. In *Mobicom '99*. ACM Press, 1999.

[15] T. DeFanti, I. Foster, M. Papka, R. Stevens, and T. Kuhfuss. Overview of the I–WAY: Wide–Area Visual Supercomputing. *The International Journal of Supercomputer Applications and High Performance Computing*, 10(2):123–130, Summer/Fall 1996.

[16] Luiz DeRose and David Padua. A MATLAB to Fortran 90 translator and its effectiveness. In *Proceedings of the 10th International Conference on Supercomputing*, May 1996.

[17] P. Dinda and D. O'Hallaron. An Evaluation of Linear Models for Host Load Prediction. In *Proceedings of the 8th IEEE Symposium on High-Performance Distributed Computing (HPDC)*. IEEE Press, 1999.

[18] A. Downey. Predicting Queue Times on Space-Sharing Parallel Computers. In *Proceedings of the International Parallel Processing Symposium*, 1997.

[19] Entropia, `http://www.entropia.com`.

[20] S. Fitzgerald, I. Foster, C. Kesselman, G. von Laszewski, W. Smith, and S. Tuecke. A Directory Service for Configuring High–Performance Distributed Computations. In *Proceedings of the Sixth IEEE Symposium on High–Performance Distributed Computing*, pages 365–375, August 1997.

[21] Network for Earthquake Engineering Simulation. `http://www.neesgrid.org/`.

[22] Search for Extraterrestrial Intelligence. `http://setiathome.ssl.berkeley.edu/`.

[23] I. Foster, J. Geisler, W. Nickless, W. Smith, and S. Tuecke. Software Infrastructure for the I–WAY Metacomputing Experiment. (To appear in *Concurrency: Practice & Experience.*).

[24] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1998.

[25] I. Foster and C. Kesselman. The Globus Toolkit. In I. Foster and C. Kesselman, editors, *The Grid: Blueprint for a New Computing Infrastructure*, pages 259–278. Morgan Kaufmann, 1998.

[26] I. Foster, C. Kesselman, C. Lee, R. Lindell, K. Nahrstedt, and A. Roy. A Distributed Resource Management Architecture that Supports Advance Reservations and Co-Allocation. In *Proceedings of the International Workshop on Quality of Service*, pages 27–36, 1999.

[27] I. Foster, C. Kesselman, and S. Tuecke. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of Supercomputer Applications*, 2001.

[28] Edgar Gabriel, Michael Resch, Thomas Beisel, and Rainer Keller. Distributed Computing in a Heterogenous Computing Environment. In *Proc. EuroPVMMPI'98*. 1998.

[29] D. Gannon, R. Bramley, M. Govindaraju, N. Mukhi, M. Yechuri, and B. Temko. A Componentized Services Architecture for Building Distributed Grid Applications. In *Proceedings of Ninth IEEE International Symposium on High Performance Distributed Computing*, Pittsburgh, Pennsylvania, August 2000.

[30] Dennis Gannon and Andrew Grimshaw. Object-based approaches. In I. Foster and C. Kesselman, editors, *The Grid: Blueprint for a New Computing Infrastructure*, pages 205–236. Morgan Kaufmann Publishers.

[31] European Data Grid. `http://www.eu-datagrid.org/`.

[32] NASA Information Power Grid. `http://www.nas.nasa.gov/About/IPG/ipg.html`.

[33] A. S. Grimshaw, W. A. Wulf, and the Legion Team. The Legion Vision of a Worldwide Virtual Computer. *Communications of the ACM*, 40(1), 1997.

[34] Grid Physics Network (GriPhyN) Project, `http://www.griphyn.com`.

[35] E. Guttman, C. Perkins, J. Veizades, and M. Day. Service Location Protocol, Version 2. IETF RFC 2165, 1998.

[36] Samuel Guyer and Calvin Lin. An annotation language for optimizing software libraries. In *Proceedings of the Second Conference on Domain-Specific Languages*, October 1999.

[37] B. Hahn. *Essential MATLAB for Scientists and Engineers.* Arnold, 1997.

[38] E. Houstis and J. Rice. Ellpack: An expert system for parallel processing of partial differential equations, 1990.

[39] T. A. Howes and M. Smith. A Scalable, Deployable Directory Service Framework for the Internet. Technical Report 95-7, Center for Information Technology Integration, Univerity of Michigan, 1995.

[40] William E. Johnston, Dennis Gannon, and Bill Nitzberg. Grids as Production Computing Environments: The Engineering Aspects of NASA's Information Power Grid. In *Proceedings of the 8th IEEE Symposium on High-Performance Distributed Computing (HPDC)*. IEEE Computer Society Press, 1999.

[41] N. H. Kapadia, J. A. B. Fortes, and C. E. Brodley. Predictive Application-Performance Modeling in a Computational Grid Environment. In *Proceedings of the 8th IEEE Symposium on High-Performance Distributed Computing (HPDC)*, August 1999.

[42] K. Kennedy, B. Broom, K. Cooper, J. Dongarra, R. Fowler, D. Gannon, L. Johnsson, J. Mellor-Crummey, and L. Torczon. Telescoping Languages: A Strategy for Automatic Generation of Scientific Problem-Solving Systems from Annotated Libraries. *Journal of Parallel and Distributed Computing*, 2001. To appear.

[43] Khoral Software. *Khoros Pro Version 2.2*, 1998.

[44] T. Kimura and H. Takemiya. Local Area Metacomputing for Multidisciplinary Problems: A Case Study for Fluid/Structure Coupled Simulation. In *Proc. Intl. Conf. on Supercomputing*, pages 145–156. 1998.

[45] M. Livny. Matchmaking: Distributed Resource Management for High Throughput Computing. In *Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing*, 1998.

[46] Miron Livny. High-throughput resource management. In I. Foster and C. Kesselman, editors, *The Grid: Blueprint for a New Computing Infrastructure*, pages 311–337. Morgan Kaufmann Publishers, 1998.

[47] Mark Lutz. *Programming Python.* O'Reilly & Associates, October 1996.

[48] P. Lyster, L. Bergman, P. Li, D. Stanfill, B. Crippe, R. Blom, C. Pardo, and D. Okaya. CASA Gigabit Supercomputing Network: CALCRUST Three–Dimensional Real–Time Multi–Dataset Rendering. In *Proceedings of Supercomputing '92*, Minneapolis, Minnesota, November 1992. (Poster session.).

[49] Vijay Menon and Keshav Pingali. A case for source-level transformations in MATLAB. In *Proceedings of the Second Conference on Domain-Specific Languages*, pages 53–65, October 1999.

[50] Vijay Menon and Keshav Pingali. High-level semantic optimization of numerical codes. In *Proceedings of the International Conference on Supercomputing 1999*, pages 434–443, 1999.

[51] Napster, `http://www.napster.com`.

[52] D. Narayanan, J. Flinn, and M. Satyanarayanan. Using History to Improve Mobile Application Adaptation. In *Proceedings of the Third Workshop on Mobile Computing Systems and Applications*, December 2000.

[53] Parabon, `http://www.parabon.com`.

[54] A. Petitet, S. Blackford, J. Dongarra, B. Ellis, Graham Fagg, K. Roche, and S. Vadhiyar. Numerical libraries and the grid: The grads experiments with scalapack. *International Journal of Supercomputer Applications*, 2001.

[55] D. Reed, C. Elford, T. Madhyastha, E. Smirni, and S. Lamm. The Next Frontier: Interactive and Closed Loop Performance Steering. In *Proceedings of the 1996 International Conference on Parallel Processing Workshop*, pages 20–31, August 1996.

[56] D. Reed and R. L. Ribler. Performance Analysis and Visualization. In I. Foster and C. Kesselman, editors, *The Grid: Blueprint for a New Computing Infrastructure*, pages 367–394. Morgan Kaufmann, 1998.

[57] R. Ribler and D. Reed. The Autopilot Performance–Directed Adaptive Control System. In *Proceedings of 11th ACM International Conference on Supercomputing—Workshop on Performance Data Mining: Automated Diagnosis, Adaption and Optimization*, Vienna, Austria, July 1997.

[58] M. Ripeanu, A. Iamnitchi, and I. Foster. Performance predictions for a numerical relativity package in grid environments. *International Journal of Supercomputer Applications*, 2001.

[59] T. Sheehan, W. Shelton, T. Pratt, P. Papadopoulos, P. LoCascio, and T. Dunigan. Locally Self Consistent Multiple Scattering Method in a Geographically Distributed Linked MPP Environment. *Parallel Computing*, 24, 1998.

[60] Steve Smale. Dynamics in General Equilibrium Theory. *American Economic Review*, 66(2):284–294, May 1976.

[61] W. Smith, I. Foster, and V. Taylor. Predicting Application Run Times Using Historical Information. In *The 4th Workshop on Job Scheduling Strategies for Parallel Processing*, 1998.

[62] H. J. Song, X. Liu, D. Jakobsen, R. Bhagwan, X. Zhang, K. Taura, and A. Chien. The MicroGrid: a scientific tool for modeling computational grids. In *SC2000*. 2000.

[63] R. Stevens, P. Woodward, T. DeFanti, and C. Catlett. From the I–WAY to the National Technology Grid. *Communications of the ACM*, 40(11):50–60, November 1997.

[64] Jaspal Subhlok, Peter Lieu, and Bruce Lowekamp. Automatic Node Selection for High Performance Applications on Networks. In *Proceedings of the Seventh ACM SIGPLAN Symposium on the Principles and Practice of Parallel Programming (PPoPP'99)*, pages 163–172. ACM Press, 1999.

[65] M. van Steen, F. Hauck, P. Homburg, and A. Tanenbaum. Location Objects in Wide-area Systems. *IEEE Communications Magazine*, pages 104–109, 1998.

[66] F. Vraalsen, R. Aydt, C. Mendes, and D. Reed. Performance Contracts: Predicting and Monitoring Grid Application Behavior. In Proceedings of The Second IEEE/ACM International Workshop on Grid Computing, November 2001.

[67] J. Waldo. The Jini Architecture for Network-centric Computing. *Communications of the ACM*, 42(7):76–82, July 1999.

[68] Carl A. Waldspurger, Tad Hogg, Bernardo A. Huberman, Jeffery O. Kephart, and W. Scott Stornetta. Spawn: A Distributed Computational Economy. *IEEE Trans. on Software Engineering*, 18(2):103–117, February 1992.

[69] Glen E. Weaver, K. S. McKinley, and Charles C. Weems. Score: A compiler representation for heterogeneous systems. In *Proceedings of the 1996 Heterogeneous Computing Workshop*, Honolulu, April 1996.

[70] R. Wolski. Dynamically Forecasting Network Performance to Support Dynamic Scheduling Using the Network Weather Service. In *Proceedings of the Sixth IEEE Symposium on High–Performance Distributed Computing*, August 1997.