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-Crumme, Dan Reed, Linda Torczon and Rich Wolski

The online version of this article can be found at:

http://hpc.sagepub.com/content/15/4/327

Published by:

$SAGE

http://www.sagepublications.com

Additional services and information for *International Journal of High Performance Computing Applications* can be found at:

Email Alerts: http://hpc.sagepub.com/cgi/alerts

Subscriptions: http://hpc.sagepub.com/subscriptions

Reprints: http://www.sagepub.com/journalsReprints.nav

Permissions: http://www.sagepub.com/journalsPermissions.nav

Citations: http://hpc.sagepub.com/content/15/4/327.refs.html

THE GrADS PROJECT: SOFTWARE SUPPORT FOR HIGH-LEVEL GRID APPLICATION DEVELOPMENT

Francine Berman[1] **Ken Kennedy**[2]
Andrew Chien[1] **Carl Kesselman**[8]
Keith Cooper[2] **John Mellor-Crummey**[2]
Jack Dongarra[3] **Dan Reed**[9]
Ian Foster[4,5] **Linda Torczon**[2]
Dennis Gannon[6] **Rich Wolski**[10]
Lennart Johnsson[7]

Summary

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, which can be used in an on-demand fashion as a problem-solving resource in many fields of human endeavor—and, in particular, 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, rather, 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. 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, the authors 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.

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, and, finally, triggering nanoactuators to remove inappropriate concentrations of effluent or other nonnative materials.

Imagine an earthquake engineering system that integrates "teleobservation" and "teleoperation" to enable researchers to control experimental tools (e.g., seismographs, cameras, 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: com-

[1]UNIVERSITY OF CALIFORNIA, SAN DIEGO
[2]RICE UNIVERSITY
[3]UNIVERSITY OF TENNESSEE, KNOXVILLE
[4]ARGONNE NATIONAL LABORATORY
[5]UNIVERSITY OF CHICAGO
[6]INDIANA UNIVERSITY
[7]UNIVERSITY OF HOUSTON
[8]UNIVERSITY OF SOUTHERN CALIFORNIA
[9]UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
[10]UNIVERSITY OF CALIFORNIA, SANTA BARBARA

"These examples illustrate what we believe will be three dominant themes in 21st-century computing: computing resources are no longer localized but, rather, 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."

puting resources are no longer localized but, rather, 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 so on. 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 National Science Foundation (NSF) Next Generation Software Program in 1999. In this paper, we present the vision and strategies underlying the GrADS effort.

Our use of the term "the Grid" is inspired by a recently published volume titled *The Grid: Blueprint for a New Computing Infrastructure* (Foster and Kesselman, 1999b), 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 (Stevens et al., 1997; Foster and Kesselman, 1999b). 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 the National Aeronautic and Space Administration (NASA) Information Power Grid (Johnston, Gannon, and Nitzberg, 1999) (http://www.nas. nasa.gov/About/IPG/ipg.html), the NSF PACI National Technology Grid (Stevens et al., 1997) and Distributed Terascale Facility, the NSF Grid Physics Network (GriPhyN) (http://www.griphyn.com), and the European Union Data Grid (http://www.eu-datagrid.org/) 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 com-

plex 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 (Lyster et al., 1992; DeFanti et al., 1996; Foster et al., 2001; Foster and Kesselman, 1999b; Gabriel et al., 1998; Kimura and Takemiya, 1998; Sheehan et al., 1998). 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 us 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 on a significant body of knowledge and technology in distributed computing, the Internet, and grid middleware. However, although 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 test beds for experimentation with the GrADS 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 using 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 (e.g., 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 (Foster and Kesselman, 1999a), NetSolve (Casanova and Dongarra, 1997), Condor (Livny, 1998, 1999), Legion (Grimshaw, Wolf, and the Legion Team, 1997; Gannon and Grimshaw, 1999), and commercial infrastructure systems have different levels of robustness and operate on different resource subsets. Over the next 10 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 runtime systems for such adaptable and "ultraportable" 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 in which the user does not know (or care) where the application might be executed. Current examples of such software platforms include Search for Extraterrestrial Intelligence (http://setiathome.ssl.berkeley.edu/) and Entropia (http://www.entropia.com) (which target largely embarrassingly parallel appli-

cations 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 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 under way for production grid systems (e.g., the NASA Information Power Grid, the NSF GriPhyN and Network for Earthquake Engineering Simulation [http://www.neesgrid.org/], and the European Union Data Grid) 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 its software development serve as important motivators 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—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 colocated 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 NSF GriPhyN project is developing a distributed analysis environment for physics experiments that will serve thousands of users (Chervenak et al., 2000). A crucial concept being pioneered by GriPhyN is virtual data, that is, 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 and 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 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 to 2001 pursued jointly between the 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 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 in this manner, 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 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: 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 postmortem analy-

"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."

sis 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 on 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 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 to assemble grid applications from a toolkit of domain-specific components. Another path allows developers to build the specialized components that form these problem-solving environment toolkits (e.g., a library for solving partial differential equations (PDEs) on computational grids) or to create new modules for their specific problem domain.

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
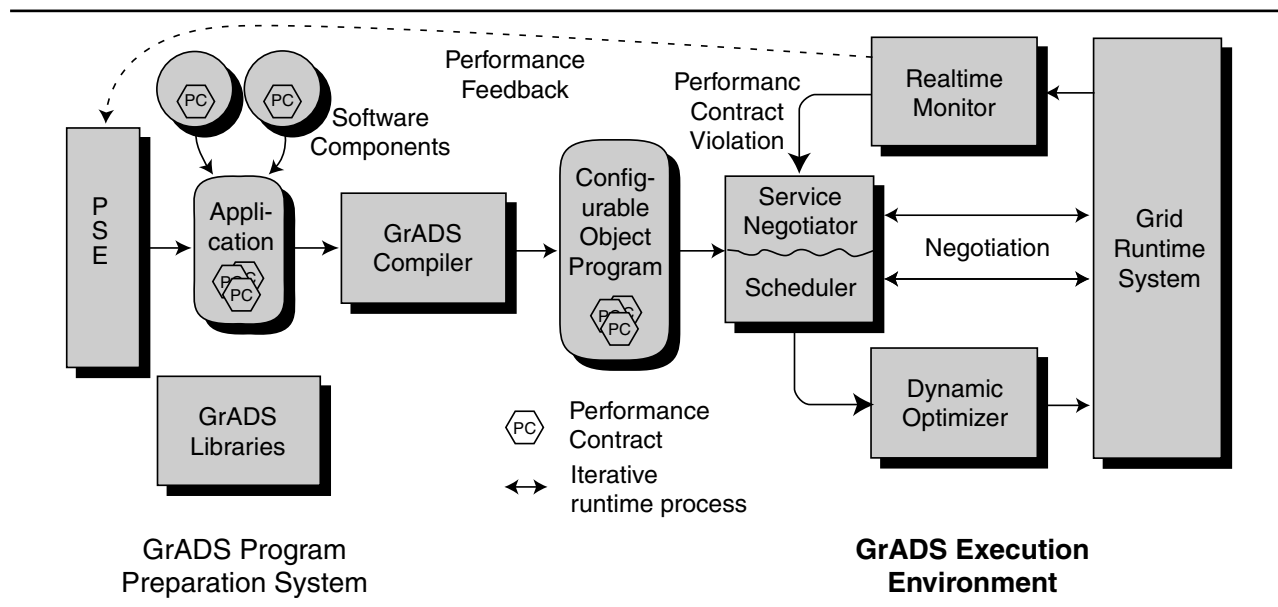

Fig. 1   GrADS preparation and execution architecture

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 in which 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 on preexisting component libraries. This approach hides grid-level details and allows the user to 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 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 system should provide several ways to construct applications. We expect that the most common approach will be to compose applications from prewritten, domain-specific library components as is done with

CCAT (Gannon et al., 2000), Khoros (Khoral Software, 1998), SciRun (Casanova et al., 1999), and NetSolve (Casanova et al., 1999). 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 written using a high-level scripting language such as Python (Lutz, 1996) or Matlab (Hahn, 1997).

To bridge the gap between these comfortable, high-level scripting languages and an efficient, grid-enabled 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 and a tool for load-time tailoring that we call the dynamic optimizer.

Telescoping languages. The telescoping-languages approach (Kennedy et al., 2001) 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 on work in high-level axiom-driven optimization (Menon and Pingali, 1999a, 1999b), call-site analysis and library routine implementation selection (Guyer and Lin, 1999), and interprocedural optimization of high-level languages (DeRose and Padua, 1996; Chauveau and Bodin, 1999).

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 on 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) (Glen et al., 1996) 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 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. Although traditional numerical libraries, such as LAPACK (Anderson et al., 1999), Ellpack (Houstis and Rice, 1990), ScaLAPACK (Blackford
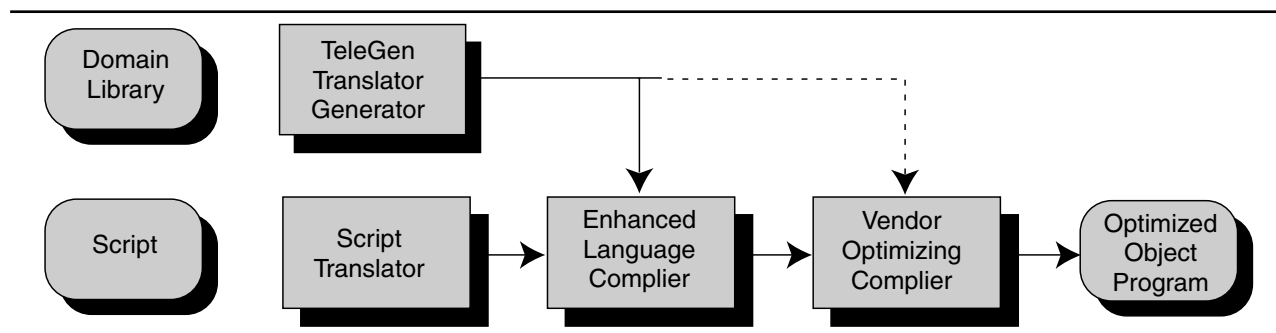
Fig. 2   Telescoping languages

et al., 1997), and PETSc (Balay et al., 2001), have brought immense benefits, the radical changes occurring in scientific computing are creating new challenges that these libraries, in their current form, cannot meet, despite the immense benefits of such traditional numerical libraries.

To address the present and future challenges in scientific computing, we are developing a new generation of Self-Adapting Numerical Software (SaNS). 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 allow the library to 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 enables 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 allow us to 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 Execution Environment for Grid Applications

As explained earlier, 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 allow 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 (Aiken et al., 2000): 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 (Foster, Kesselman, and Tuecke, 2001).

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 (Foster and Kesselman, 1999a), due to 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 coallocation 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 between cooperating services and resources. Issues of security, resource access, and so on can be relegated to Globus—and/or to complementary industrial standards and trends such as Jini (Waldo, 1999) and the emerging peer-to-peer technology base (as is being pioneered by companies such as Entropia, CDDB [http://www.cddb.com] [Casanova and Dongarra, 1997], Napster [http://www.napster.com], and Parabon [http://www.parabon.com]). 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 feedback useful to the grid protocols and services R&D community.

In the following, we expand on 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 (Narayanan, Flinn, and Satyanarayanan, 2000), estimates of nonobservable system properties, and predictions of future states (Wolski, 1997; Dinda and O'Hallaron, 1999; Smith, Foster, and Taylor, 1998; Downey, 1997; Narayanan, Flinn, and Satyanarayanan, 2000; Kapadia, Fortes, and Brodley, 1999). Our goals in the GrADS project are as follows: develop an integrated framework in which these many different types of information can be used in a coordinated and uniform fashion; conduct a broad exploration of how different sorts of information can be produced and used; and produce a set of effective techniques for information collection, analysis, and application.

In previous work, we established frameworks for providing uniform access to, and indexing of, diverse information sources (the Globus MDS) (Fitzgerald et al., 1997; Czajkowski et al., 2001); for collecting experimental data and using this data to generate forecasts of future state (the Network Weather Service) (Wolski, 1997); and for structuring networks of sensors and transformers to support adaptive control (Autopilot) (Reed and Ribler, 1999). 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 (van Steen et al., 1998; Czerwinski et al., 1999; Guttman et al., 1998; Howes and Smith, 1995). 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 (Czajkowski et al., 1998), which provides secure remote access and reservation mechanisms (Foster et al., 1999), we are developing new coreservation and coallocation algorithms capable of dealing with resources with dynamic and probabilistic properties, integrating performance contracts (see next paragraph) into resource reservation and resource operations, integrating traditional quality of service methods into resource management frameworks, and mapping 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, multidimensional requirements and performance data between resource providers and consumers. To that end, we are investigating the design of a language of performance contracts to enable dynamic negotiation between 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 (Livny, 1998; see also Subhlok, Lieu, Lowekamp, 1999).

Building on this framework, we will investigate more dynamic resource-brokering mechanisms based on the use of economic models (e.g., bidding, cost negotiation, 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 (Waldspurger et al., 1992;

Smale, 1976). Because 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 between applications, intermediate brokering functions, and resources. The third area in which we are working is building on this framework to construct 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 allow it to 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 (Ribler and Reed, 1997; Reed et al., 1996; Reed and Ribler, 1999) 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 allow the compiler, scheduler, runtime system, and other components to extract, cooperatively and nonintrusively, pertinent information from the running application.

Our preliminary experiments with performance contracts (Vraalsen et al., 2001) indicate that this is a fruitful approach. Using Autopilot's fuzzy logic decision procedures, it was possible to detect local perturbations in pro-

cessor 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 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 toward the development of such design methodologies, we are developing and deploying two major test beds 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 test beds and tool suites are referred to as the MicroGrid and MacroGrid test beds. 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 test beds, hence allowing comparative studies.

The MicroGrid test bed, 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 test bed for grid experiments. Major challenges include the following:

- *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 simu-

lation 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 (Song et al., 2000), 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 test bed. This test bed provides a more controlled environment, and likely a much higher degree of instrumentation and data capture, than is possible in typical grid environments. This test bed is being used for the initial application 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 to understand 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, test bed 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

"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."

solver (Petitet et al., 2001) and a grid-enabled version of Cactus (Allen et al., 2000; Allen et al., 2001; Ripeanu, Iamnitchi, and Foster, 2001), 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 between algorithms, compilers, tools, and libraries.

The test bed 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, the Network Weather Service, 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 test bed 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, test bed, and application evaluation thrusts are linked in 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 constructed two major test beds (MicroGrid and 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 such as 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

BIOGRAPHIES

Francine Berman is director of the San Diego Supercomputer Center, director of the National Partnership for Advanced Computational Infrastructure, professor of computer science and engineering at the University of California, San Diego (UCSD), fellow of the Association for Computing Machinery, and director of the Grid Computing Laboratory at UCSD. Her research interests over the past two decades have focused on parallel and distributed computation, in particular the areas of programming environments, tools, and models that support high-performance computing. Her current research focuses on the development of performance-oriented software, models, and applications for networked heterogeneous distributed resources, also known as computational Grids or Metasystems. She has served on numerous boards and program and conference committees, providing expertise in the areas of parallel computing and grid computing. She received a B.A. from the University of California, Los Angeles, in 1973. She obtained an M.S. and a Ph.D. in computer science from the University of Washington in 1976 and 1979, respectively.

Andrew Chien is the Science Applications International Corporation chair professor in the Department of Computer Science and Engineering at the University of California, San Diego. He is also the chief technology officer and cofounder of Entropia Inc. He received his undergraduate, master's, and doctoral degrees from the Massachusetts Institute of Technology in 1984,

1987, and 1990, respectively. From 1990 to 1998, he was a faculty member in the Department of Computer Science at the University of Illinois and a senior research scientist in the National Center for Supercomputing Applications. He is also a recipient of a 1994 National Science Foundation Young Investigator Award, a 1995 C. W. Gear Outstanding Faculty Award, and a 1996 Xerox Outstanding Research Award.

Keith Cooper is a professor in the Department of Computer Science at Rice University. His research has focused on techniques for compiler-based optimization and code generation. This has led him into work on interprocedural analysis and optimization, code generation, optimization methods for uniprocessor machines, and the application of classical optimization techniques in very high speed integrated circuit hardware description language (VHDL) compilation. His current interests include adaptive compilers, binary translation, techniques for reducing power consumption, and code generation for aggressive microprocessors. He is a coauthor of *Engineering a Compiler* (with Linda Torczon), which will be published in 2002 by Morgan-Kaufmann.

Jack Dongarra holds an appointment as university distinguished professor of computer science in the Computer Science Department at the University of Tennessee and is an adjunct R&D participant in the Computer Science and Mathematics Division at Oak Ridge National Laboratory and an adjunct professor in computer science at Rice University. He specializes in numerical algorithms in linear algebra, parallel computing, use of advanced-computer architectures, programming methodology, and tools for parallel computers. His research includes the development, testing, and documentation of high-quality mathematical software. He has contributed to the design and implementation of the following open source software packages and systems: EISPACK, LINPACK, the BLAS, LAPACK, ScaLAPACK, Netlib, PVM, MPI, NetSolve, Top500, ATLAS, and PAPI. He has published approximately 200 articles, papers, reports, and technical memoranda, and he is coauthor of several books. He is a fellow of the American Association for the Advancement of Science, the Association for Computing Machinery, and the Institute of Electrical and Electronics Engineers, and a member of the National Academy of Engineering.

Ian Foster is senior scientist and associate director of the Mathematics and Computer Science Division at Argonne National Laboratory, professor of computer science at the University of Chicago, and senior fellow in the Argonne/University of Chicago Computation Institute. He has published four books and more than 100 papers and technical reports in parallel and distributed processing, software engineering, and computational science. He currently coleads the Globus project with Carl Kesselman of the University of Southern California/Information Sciences Institute, which won the 1997 Global Information Infrastructure Next Generation Award and provides protocols and services used by many distributed computing projects worldwide. He cofounded the influential Global Grid Forum

and recently coedited a book on this topic, published by Morgan-Kaufmann, titled *The Grid: Blueprint for a New Computing Infrastructure*.

Dennis Gannon is a professor in the department of computer science at Indiana University, which he also chairs. His previous positions include the Department of Computer Science at Purdue University and Center for Supercomputer Research and Development at the University of Illinois. He is a founding member of the Department of Energy 2000 Common Component Architecture software tools group and the National Center for Supercomputing Applications Alliance, where he is chief computer scientist. He also helped found the Java Grande Forum. His current research interests involve the construction of distributed applications based on software component technology, the integration of parallel and distributed programming systems, the design of problem-solving workbenches, and distributed grid services. He is also the science directory for the Pervasive Technology Labs at Indiana University.

Lennart Johnsson is Hugh Roy and Lillie Cranz Cullen Distinguished Professor of Computer Science, Mathematics and Electrical and Computer Engineering, College of Natural Sciences and Mathematics and College of Engineering, University of Houston.

Ken Kennedy is the Ann and John Doerr professor of computational engineering and director of the Center for High Performance Software Research (HiPerSoft) at Rice University. He is a fellow of the Institute of Electrical and Electronics Engineers (IEEE), the Association for Computing Machinery (ACM), and the American Association for the Advancement of Science, and has been a member of the National Academy of Engineering since 1990. From 1997 to 1999, he served as cochair of the President's Information Technology Advisory Committee (PITAC). For his leadership in producing the PITAC report on funding of information technology research, he received the Computing Research Association Distinguished Service Award (1999) and the RCI Seymour Cray HPCC Industry Recognition Award (1999). He has published more than 150 technical articles and supervised 34 Ph.D. dissertations on programming support software for high-performance computer systems. In recognition of his contributions to software for high performance computation, he received the 1995 W. Wallace McDowell Award, the highest research award of the IEEE Computer Society. In 1999, he was named the third recipient of the ACM SIGPLAN Programming Languages Achievement Award.

Carl Kesselman is a senior project leader at the Information Sciences Institute and a research associate professor of computer science, both at the University of Southern California. He is also a visiting associate in computer science at the California Institute of Technology. He received a Ph.D. in computer science from the University of California, Los Angeles, in 1991. He currently coleads the Globus project with Ian Foster of Argonne National Laboratory/University of Chicago, which won the 1997 Global Information Infrastructure Next Genera-

tion Award and provides protocols and services used by many distributed computing projects worldwide. He recently coedited a book on this topic, published by Morgan-Kaufmann, titled *The Grid: Blueprint for a New Computing Infrastructure*.

John Mellor-Crummey earned a B.S.E. degree magna cum laude in electrical engineering and computer science from Princeton University in 1984, and M.S. (1986) and Ph.D. (1989) degrees in computer science from the University of Rochester. In 1989, he joined the Department of Computer Science and the Center for Research on Parallel Computation (CRPC) at Rice University, where he currently holds the rank of senior faculty fellow. He was a member of the technical steering committee of the CRPC, a National Science Foundation Science and Technology Center, and is currently a member of the executive committee for the Los Alamos Science Institute—a joint venture between Los Alamos National Laboratory and Rice University. In recent years, his research has focused on compilers, runtime libraries, and programming environments for parallel processing. He is most widely known for multiprocessor synchronization algorithms he developed with Michael Scott (University of Rochester). Over the past several years, he has led the dHPF compiler project, a long-term research effort that has focused on developing compiler and tool technology to support construction of efficient data-parallel scientific applications. He is a member of Tau Beta Pi and Phi Beta Kappa.

Dan Reed is director of the National Center for Supercomputing Applications and the National Computational Science Alliance, one of two National Science Foundation PACI partnerships. He is also an Edward William Gutgsell and Jane Marr Gutgsell professor at the University of Illinois at Urbana-Champaign. He received a B.S. (summa cum laude) in computer science from the University of Missouri at Rolla in 1978 and an M.S. and a Ph.D., also in computer science, from Purdue University in 1980 and 1983, respectively. He is a member of several national collaborations, including the Center for Grid Application Development Software, the Department of Energy Accelerated Strategic Computing Initiative and the Scientific Discovery through Scientific Computing program, and the Los Alamos Computer Science Institute. He also serves on the board of directors of the Computing Research Association.

Linda Torczon is a research scientist in the Department of Computer Science at Rice University. She is the executive director of the Los Alamos Computer Science Institute and the Grid Application Development Software project funded by the National Science Foundation (NSF). She also served as executive director of the Center for Research on Parallel Computation, an NSF Science and Technology Center, from 1990 to 2000. Her research interests include code generation, adaptive compilation, interprocedural data flow analysis and optimization, and programming environments. Techniques that she developed are widely used in industrial and research compilers. She is the co-author of *Engineering a Compiler* (with Keith Cooper), which will be published in 2002 by Morgan-Kaufmann.

Rich Wolski is an assistant professor of computer science at the University of California, Santa Barbara. His research interests include computational grid computing, distributed computing, scheduling, and resource allocation. In addition to the EveryWare project, he leads the Network Weather Service project, which focuses on online prediction of resource performance, and the G-Commerce project, which focuses on computational economies for the Grid.

REFERENCES

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

Allen, G., Angulo, D., Foster, I., Lanfermann, G., Liu, C., Radke, T., Seidel, E., and Shalf, J. 2001. The Cactus Worm: Experiments with dynamic resource discovery and allocation in a grid environment. *International Journal of High Performance Computer Applications* 15:345-358.

Allen, G., Benger, W., Goodale, T., Hege, H., Lanfermann, G., Merzky, A., Radke, T., and Seidel, E. 2000. The Cactus code: A problem solving environment for the grid. In *Proceedings of the Ninth IEEE International Symposium on High Performance Distributed Computing*, 253-260. Pittsburgh, PA: IEEE Computer Society Press.

Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Du Croz, J., Greenbaum, A., Hammarling, S., McKenney, A., and Sorensen, D. 1999. *LAPACK Users' Guide*. 3rd ed. Philadelphia: SIAM.

Balay, S., Gropp, W. D., McInnes, L. C., and Smith, B. F. 2001. PETSc users' manual. Technical Report ANL-95/11, Revision 2.1.0, Argonne National Laboratory.

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

Casanova, H., and Dongarra, J. 1997. NetSolve: A network-enabled server for solving computational science problems. *International Journal of High Performance Computing Applications* 11:212-223.

Casanova, H., Dongarra, J., Johnson, C., and Miller, M. 1999. Application-specific tools. In *The Grid: Blueprint for a New Computing Infrastructure*, edited by I. Foster and C. Kesselman, 159-180. San Francisco: Morgan Kaufmann.

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

Chervenak, A., Foster, I., Kesselman, C., Salisbury, C., and Tuecke, S. 2000. The Data Grid: Towards an architecture for the distributed management and analysis of large scientific data sets. *Journal of Network and Computer Applications* 23:187-200.

Czajkowski, K., Fitzgerald, S., Foster, I., and Kesselman, C. 2001. Grid information services for distributed resource

sharing. In *Proceedings of the 10th IEEE Symposium on High-Performance Distributed Computing (HPDC)*. IEEE Computer Society Press.

Czajkowski, K., Foster, I., Karonis, N., Kesselman, C., Martin, S., Smith, W., and Tuecke, S. 1998. A resource management architecture for metacomputing systems. In *Proceedings of the Fourth Workshop on Job Scheduling Strategies for Parallel Processing*.

Czerwinski, S. E., Zhao, B. Y., Hodes, T. D., Joseph, A. D., and Katz, R. H. 1999. An architecture for a secure service discovery service. In *Mobicom '99*. ACM Press.

DeFanti, T., Foster, I., Papka, M., Stevens, R., and Kuhfuss, T. 1996. Overview of the I-WAY: Wide-area visual supercomputing. *International Journal of High Performance Computing Applications* 10:123-130.

DeRose, L., and Padua, D. 1996. A MATLAB to Fortran 90 translator and its effectiveness. In *Proceedings of the 10th International Conference on Supercomputing*, May.

Dinda, P., and O'Hallaron, D. 1999. An evaluation of linear models for host load prediction. In *Proceedings of the 8th IEEE Symposium on High-Performance Distributed Computing (HPDC)*. IEEE Computer Society Press.

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

Fitzgerald, S., Foster, I., Kesselman, C., von Laszewski, G., Smith, W., and Tuecke, S. 1997. A directory service for configuring high-performance distributed computations. In *Proceedings of the Sixth IEEE Symposium on High-Performance Distributed Computing*, pp. 365-375.

Foster, I., Geisler, J., Nickless, W., Smith, W., and Tuecke, S. 1998. Software infrastructure for the I-WAY metacomputing experiment. *Concurrency: Practice and Experience* 10(7):567-581.

Foster, I., and Kesselman, C. 1999a. The Globus Toolkit. In *The Grid: Blueprint for a New Computing Infrastructure*, edited by I. Foster and C. Kesselman, 259-278. San Francisco: Morgan Kaufmann.

Foster, I., and Kesselman, C. 1999b. *The Grid: Blueprint for a New Computing Infrastructure*. San Francisco: Morgan Kaufmann.

Foster, I., Kesselman, C., Lee, C., Lindell, R., Nahrstedt, K., and Roy, A. 1999. A distributed resource management architecture that supports advance reservations and co-allocation. In *Proceedings of the International Workshop on Quality of Service*, pp. 27-36.

Foster, I., Kesselman, C., and Tuecke, S. 2001. The anatomy of the Grid: Enabling scalable virtual organizations. *International Journal of High-Performance Computing Applications* 15:200-220.

Gabriel, E., Resch, M., Beisel, T., and Keller, R. 1998. Distributed computing in a heterogenous computing environment. In *Proceedings of EuroPVMMPI'98*.

Gannon, D., Bramley, R., Govindaraju, M., Mukhi, N., Yechuri, M., and Temko, B. 2000. A componentized services architecture for building distributed grid applications. In *Proceed-*

ings of Ninth IEEE International Symposium on High Performance Distributed Computing, Pittsburgh, PA.

Gannon, D., and Grimshaw, A. 1999. Object-based approaches. *The Grid: Blueprint for a New Computing Infrastructure*, edited by I. Foster and C. Kesselman, 205-236. San Francisco: Morgan Kaufmann.

Grimshaw, A. S., Wulf, W. A., and the Legion Team. 1997. The legion vision of a worldwide virtual computer. *Communications of the ACM* 40(1):39-45.

Guttman, E., Perkins, C., Veizades, J., and Day, M. 1998. *Service Location Protocol Version 2*. IETF RFC 2165. Available: http://www.ietf.org/rfc/rfc2165.txt.

Guyer, S., and Lin, C. 1999. An annotation language for optimizing software libraries. In *Proceedings of the Second Conference on Domain-Specific Languages*, October.

Hahn, B. 1997. *Essential MATLAB for Scientists and Engineers*. London: Arnold.

Houstis, E., and Rice, J. 1990. Ellpack: An expert system for parallel processing of partial differential equations. In *Intelligent Mathematical Software Systems*, E. N. Houstis et al., eds., 253-260. Holland: Elsevier Science Publishers.

Howes, T. A., and Smith, M. 1995. A scalable, deployable directory service framework for the Internet. Technical Report 95-7, Center for Information Technology Integration, University of Michigan.

Johnston, W. E., Gannon, D., and Nitzberg, B. 1999. 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.

Kapadia, N. H., Fortes, J.A.B., and Brodley, C. E. 1999. Predictive application-performance modeling in a computational grid environment. In *Proceedings of the 8th IEEE Symposium on High-Performance Distributed Computing (HPDC)*, August.

Kennedy, K., Broom, B., Cooper, K., Dongarra, J., Fowler, R., Gannon, D., Johnsson, L., Mellor-Crummey, J., and Torczon, L. 2001. Telescoping languages: A strategy for automatic generation of scientific problem-solving systems from annotated libraries. *Journal of Parallel and Distributed Computing*.

Khoros Pro Version 2.2. 1998. Khoral Software.

Kimura, T., and Takemiya, H. 1998. Local area metacomputing for multidisciplinary problems: A case study for fluid/structure coupled simulation. In *Proceedings of the International Conference on Supercomputing*, pp. 145-156.

Livny, M. 1998. Matchmaking: Distributed resource management for high throughput computing. In *Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing*.

Livny, M. 1999. High-throughput resource management. In *The Grid: Blueprint for a New Computing Infrastructure*, edited by I. Foster and C. Kesselman, 311-337. San Francisco: Morgan Kaufmann.

Lutz, M. 1996. *Programming Python*. Sebastopol, CA: O'Reilly & Associates.

Lyster, P., Bergman, L., Li, P., Stanfill, D., Crippe, B., Blom, R., Pardo, C., and Okaya, D. 1992. CASA gigabit super-computing network: CALCRUST three-dimensional real-time multi-dataset rendering. In *Proceedings of Super-computing '92*, Minneapolis, MN, November (poster session).

Menon, V., and Pingali, K. 1999a. A case for source-level trans-formations in MATLAB. In *Proceedings of the Second Con-ference on Domain-Specific Languages*, October, pp. 53-65.

Menon, V., and Pingali, K. 1999b. High-level semantic optimi-zation of numerical codes. In *Proceedings of the Interna-tional Conference on Supercomputing*, pp. 434-443.

Narayanan, D., Flinn, J., and Satyanarayanan, M. 2000. Using history to improve mobile application adaptation. In *Pro-ceedings of the Third Workshop on Mobile Computing Sys-tems and Applications*, December.

Petitet, A., Blackford, S., Dongarra, J., Ellis, B., Fagg, G., Roche, K., and Vadhiyar, S. 2001. Numerical libraries and the Grid. *International Journal of Supercomputer Applica-tions* 15:359-374.

Reed, D., Elford, C., Madhyastha, T., Smirni, E., and Lamm, S. 1996. The next frontier: Interactive and closed loop perfor-mance steering. In *Proceedings of the 1996 International Conference on Parallel Processing Workshop*, August, pp. 20-31.

Reed, D., and Ribler, R. L. 1999. Performance analysis and visu-alization. In *The Grid: Blueprint for a New Computing Infra-structure*, edited by I. Foster and C. Kesselman, 367-394. San Francisco: Morgan Kaufmann.

Ribler, R., and Reed, D. 1997. The Autopilot performance-directed adaptive control system. In *Proceedings of the 11th ACM International Conference on Supercomputing—Workshop on Performance Data Mining: Automated Diag-nosis, Adaption and Optimization*, Vienna, Austria, July.

Ripeanu, M., Iamnitchi, A., and Foster, I. 2001. Performance predictions for a numerical relativity package in grid envi-ronments. *International Journal of High Performance Com-puting Applications* 15:375-387.

Sheehan, T., Shelton, W., Pratt, T., Papadopoulos, P., LoCascio, P., and Dunigan, T. 1998. Locally self consistent multiple scattering method in a geographically distributed linked MPP environment. *Parallel Computing* 24:1827-1846.

Smale, S. 1976. Dynamics in general equilibrium theory. *Amer-ican Economic Review* 66:284-294.

Smith, W., Foster, I., and Taylor, V. 1998. Predicting application run times using historical information. In *4th Workshop on Job Scheduling Strategies for Parallel Processing*.

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

Stevens, R., Woodward, P., DeFanti, T., and Catlett, C. 1997. From the I-WAY to the national technology Grid. *Communi-cations of the ACM* 40 (11): 50-60.

Subhlok, J., Lieu, P., and Lowekamp, B. 1999. 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)*, 163-172. ACM Press.

van Steen, M., Hauck, F., Homburg, P., and Tanenbaum, A. 1998. Location objects in wide-area systems. *IEEE Commu-nications Magazine*, pp. 104-109.

Vraalsen, F., Aydt, R., Mendes, C., and Reed, D. 2001. Perfor-mance contracts: Predicting and monitoring grid application behavior. In *Proceedings of the Second IEEE/ACM Interna-tional Workshop on Grid Computing*, November.

Waldo, J. 1999. The Jini architecture for network-centric com-puting. *Communications of the ACM* 42 (7): 76-82.

Waldspurger, C. A., Hogg, T., Huberman, B. A., Kephart, J. O., and Stornetta, W. S. 1992. Spawn: A distributed computa-tional economy. *IEEE Transactions on Software Engi-neering* 18:103-117.

Weaver, G. E., McKinley, K. S., and Weems, C. C. 1996. Score: A compiler representation for heterogeneous systems. In *Proceedings of the 1996 Heterogeneous Computing Work-shop*, Honolulu, HI, April.

Wolski, R. 1997. Dynamically forecasting network perfor-mance to support dynamic scheduling using the Network Weather Service. In *Proceedings of the Sixth IEEE Sympo-sium on High-Performance Distributed Computing*, August.