## D Project Description

The Grid [32]—the use of the global information infrastructure as an active computational resource to solve real-world problems—is gaining wide acceptance as the next step beyond the Internet. It promises on-demand access to and integration of distributed data archives, scientific instruments, computing resources, and advanced services in a way that is more flexible and powerful than has been previously possible. As a result, the Grid is becoming a critical tool within many scientific communities and is being adopted by a growing range of industries and commercial endeavors. This rise to prominence is being supported by ambitious research programs in Grid middleware that target key technical challenges, such as security, resource discovery, and resource management, and by projects such as the NSF TeraGrid and NSF Middleware Initiative (NMI) that are creating hardware and software infrastructure.

Impressive Grid applications have been demonstrated (e.g. [28, 32, 37, 49, 78, 19, 6, 4, 16]), and major application frameworks are being constructed (e.g. the Grid Physics Network (GriPhyN) [39, 18] and the National Earthquake Engineering Simulation Grid (NEESgrid) [63, 72]). Still, because Grid program development is complex, it remains the exclusive province of large teams of specialists. Before the Grid can realize its true potential for both science and industry, major research advances are needed in programming models and tools for developing Grid applications and services.

Recognizing this need, we initiated the *Grid Application Development Software (GrADS) Project* to conduct fundamental research on software tools, programming models, and run-time services for scientific Grid computing. Over the past three years, this NSF-supported research has produced the foundation for an evolving compilation and execution infrastructure, *GrADSoft*, that we have used to conduct a range of experiments with several applications [64, 73, 20, 24, 57], thus validating the basic GrADS approach while exposing a number of remaining challenges. Principal among these critical problems are the difficulty of quickly and scalably identifying appropriate resources and the need for more powerful programming abstractions.

To address these issues, we propose the *Virtual Grid Application Development Software (VGrADS) Project*, a five-year research effort that will substantively broaden and deepen both the research agenda and technology base established under GrADS. VGrADS will develop new strategies that reduce development effort for current classes of Grid services and applications and enable development of new services and applications that realize the Grid's full power. By design, these strategies will scale to handle larger Grid configurations and meet increased user demands.

A key lesson from GrADS is that the complexity of the Grid resource environment induces complexity in software development and execution. Resource heterogeneity, dynamically fluctuating loads, and the interaction between local users and resource policies substantially complicate the development process. In addition, the number and diversity of Grid resources make effective resource selection challenging. To obtain acceptable resource utilization and application performance, expert programmers currently must develop sophisticated, *ad hoc* heuristics that require much tuning and debugging to be effective. To ease this problem, VGrADS will adopt the concept of *virtual grid (vgrid) architectures* as a fundamental organizing principle. Vgrids cleanly separate high-level programming tools, applications, and services from the complexity of dynamic Grid scheduling and resource management. This approach is analogous to one that has proven effective in sequential and parallel computing contexts, where optimizations target abstract uniprocessors and multiprocessors rather than the physical resources themselves. The same concept will form the basis of our approach to simplifying the task of Grid application development. Creating, refining, and implementing successful vgrids requires a large-scale, integrated attack at all levels of the system hierarchy.

As depicted in Figure 1, VGrADS will abstract the Grid into (1) physical resources (e.g. data archives, computing systems, and instruments); (2) abstract resource classes with specified attributes, formed by aggregating and conditioning physical resources; (3) vgrids, composed of components from abstract classes; and, finally, (4) a set of abstract programming models and development tools that target vgrids as application- and service-visible execution interfaces.

VGrADS research will focus on two major areas: execution environments and programming tools. *Execution environment* research will explore the synthesis, coordination, and measurement of vgrids, leveraging, where possible, emerging Grid services. The goals of this work are to explore (1) virtualization of resource and Grid service aggregates; (2) intelligent, rapid resource selection and management in complex, heterogeneous environments; (3) performance measurement and tuning to achieve high individual application performance; and (4) fault-resilience through replication and intermediate program state management.

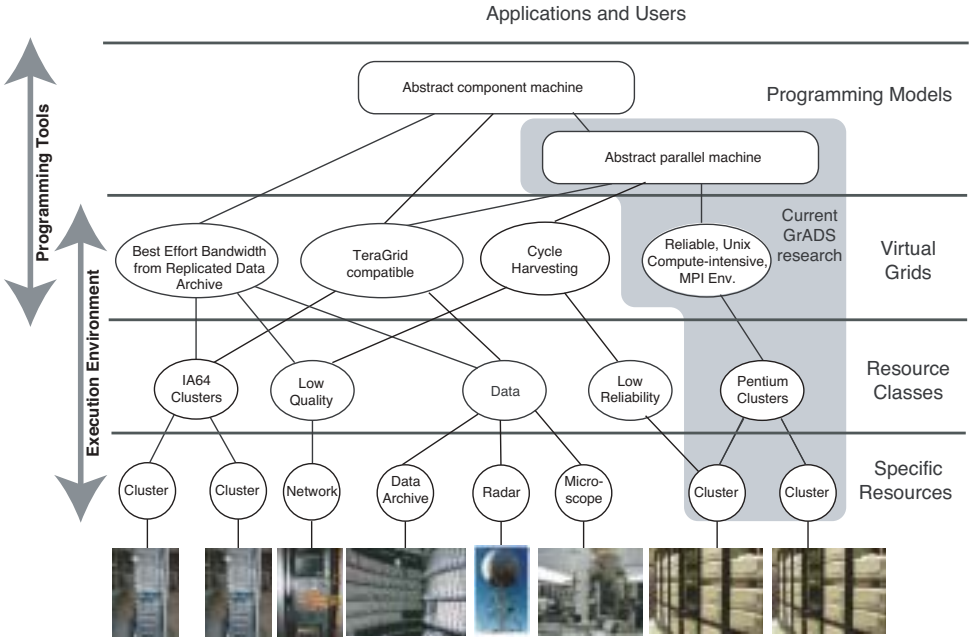


Figure 1: VGrADS virtual machine layers. The shaded region in Figure 1 represents the subset of Grid abstractions that have been the subject of GrADS research to date.

The resulting system will enable the nimble adaptation of applications to changing Grid conditions.

In *execution system* design, fundamental Grid components (resources or well-defined Grid services) are aggregated into *resource abstraction classes*. The resource selection and management infrastructure culls resources from these classes into vgrid structures that support the requirements of the application through higher-level programming tools. Thus, resource abstraction classes organize the diversity of resources that are available, and vgrids implement structure (specified by applications' needs) on those resource classes. To allow adaptation as Grid conditions change, the execution system abstractions include facilities to monitor application and resource performance at all levels. Finally, schedulers (based on abstracted resources) adapt and control application execution based on performance as measured and forecast by the infrastructure.

*Programming tools* research will explore the mapping of two distinct, high-level programming models to vgrids. The *abstract parallel machine* model treats a computation as a collection of parallel tasks without concern for mapping that computation to the actual hardware. The *abstract component machine* model, on the other hand, represents a computation as a (possibly dynamic) graph of component invocations with specific data dependencies. In this model, applications and services might be high-level scripts that invoke operations from a component integration framework. The VGrADS execution system, working on behalf of the application, will use vgrids to instantiate both of these programming models.

By carefully formulating layers of abstraction to enable efficient interaction, VGrADS will dramatically broaden the scope and scale of applications that can exploit Grid capabilities. By leveraging abstraction to improve ease of use, VGrADS will increase the number of programmers who can create Grid applications. By implementing those abstractions efficiently and portably, VGrADS will expand the computational environments where those applications can succeed. By fostering research, education, and technology transfer, VGrADS will change the way scientists, engineers, and other professionals solve their everyday problems.

### D.1 Results from Prior NSF Work (GrADS Project)

Since 1999, the GrADS project [NSF award 9975020, "Next Generation Software: Grid Application Development Software (GrADS)", $7,278,015, Oct 1999-Sep 2004; NSF award 0103759, "NGS: GrADS: Efficient Script-Based Application Development for Networked High Performance Computing Environments", $1,300,000, Oct 2002-Sep 2003] has conducted fundamental research on the technologies needed to make the Grid usable as a computationally-rich problem-solving system [48, 13, 27]. To date, GrADS has produced a coherent vision for an execution environment and application development tools that will make it far easier to produce Grid-ready applications in the future. Figure 2 illustrates the program development structure that

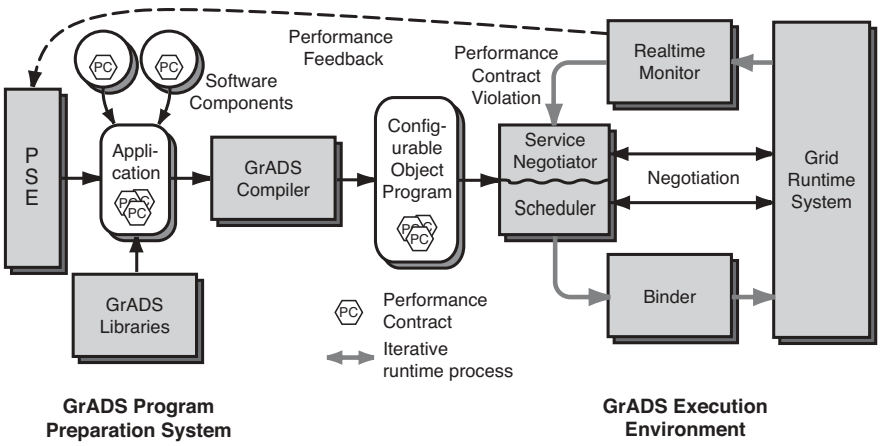


Figure 2: GrADS Program Preparation and Execution Architecture

has been pioneered by the GrADS project. In the GrADSoft architecture, the discrete steps of application creation, compilation, execution, and post-mortem analysis have been replaced with a continuous process of adapting applications to an evolving set of Grid resources and problem-specific requirements.

Two key concepts are central to this approach. First, applications are encapsulated as *configurable object programs (COPs)*, which can be optimized rapidly for execution on a specific collection of Grid resources. In the current GrADS system, a configurable object program includes: (1) *code* for the application (e.g. an MPI program); (2) a *mapper*, which determines how to map the constituent application tasks to a given set of resources; and (3) an executable *performance model*, which uses the mapper to estimate the application's performance on a set of resources. Second, the system relies upon *performance contracts* that specify the expected performance of modules as a function of available resources. The GrADS research program has studied both the execution and program preparation components of this architecture.

***GrADS Execution Environment.*** The right side of Figure 2 depicts actions when a COP is delivered to the execution environment. The GrADSoft infrastructure first determines which resources are available and secures an appropriate subset for the application. Using annotations from performance contracts and the COP's mapper and performance model, service negotiators broker the allocation and scheduling of module components on Grid resources. Next, the infrastructure invokes the *binder* to tailor the COP to the available resources. The binder also inserts sensors and actuators to let the performance monitor control application execution. The *launcher* (not shown) then presents the tailored COP to the run-time system for execution.

During program execution, a real-time monitor tracks program behavior and validates observed behavior against performance contract expectations. If a performance contract is violated, the monitor responds by interrupting execution through an actuator, triggering several possible actions. The actuator can invoke the binder with more data (from performance monitoring) to improve 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 COP for it. Dynamic forecasts of resource performance and Grid capacity reduce renegotiation overhead. Thus, this closed loop system ensures that execution of the application meets the specifications of its performance contracts in the constantly changing Grid environment.

***GrADS Program Preparation System.*** The left side of Figure 2 depicts the tools used to construct COPs. GrADSoft supports two paths for application development. In the first, the developer uses high-level problem-solving environments (PSEs) or services to assemble Grid applications from a toolkit of domain-specific components. In the second, the developer builds the specialized components that form these PSE toolkits (e.g. a library for solving PDEs) or creates new modules for their specific problem domain.

The program preparation system provides tools to assist in either scenario. These tools automatically construct performance models and mappers. They integrate modules written in derivatives of standard languages with Grid-specific extensions (e.g. data or task distribution primitives) into larger components, libraries, and applications with a coordination language. They create malleable modules, annotated with

information about their resource needs and predicted performance for a variety of resource configurations.

The focus of this research is on tools to free the user from many of the low-level concerns that arise in Grid programming using the GrADSoft infrastructure, and to permit the user to concentrate on high-level design and performance tuning for the heterogeneous distributed computing environment.

***GrADS Project Status*** GrADS has realized several important research achievements. First, we have adapted six (mostly kernel) applications for Grid execution. These applications were chosen to drive the research rather than as exemplars of ideal Grid applications. The applications include: (1) a version of the LINPACK benchmark, which was chosen as an exemplar of a tightly coupled application that amplifies the importance of good scheduling and performance modeling [65, 64]: (2) the *Cactus* numerical relativity toolkit, which was used to study scheduling and migration of large computational components in problem-solving systems [74, 4, 3, 73]; (3) the gene sequence matching application *FASTA*, which has driven the application of the GrADS scheduling approach to problems requiring fixed-location resources such as databases; (4) a general class of iterative stencil applications, which we used to develop and demonstrate a novel decoupled scheduling strategy for GrADS [24, 26, 25] as well as for exploring rescheduling [79, 80]; (5) GrADSAT, a satisfiability application for circuit design, which requires dynamic scheduling of new tasks at run time [20]; and (6) an HPF version of the mesh generation benchmark *tomcatv*, which was used to explore task graph clustering as a high-level mapping strategy [57]. (A recent version of our HPF compiler, developed for the POEMS project [1, 2], produces SPMD task graphs.) These application studies led to many insights and numerous publications. The Cactus work led to a Gordon Bell prize at SC'01.

Second, we designed and constructed a prototype execution system based on the preliminary application studies. It takes a COP and matches it to available resources [83, 84] using an optimization procedure that calls the application performance model as an objective function [25, 100]. Next, it launches the COP on the Grid via Globus and monitors execution to check if performance is acceptable [90, 89, 53, 60]. The GrADSoft prototype was demonstrated at SC'02 on FASTA and GrADSAT (described above). All six applications currently work in this prototype. The prototype does not yet handle contract violations or perform rescheduling; both will be incorporated by SC'03.

Third, we have developed several prototype tools for constructing GrADS applications. These include a tool to import Condor ClassADs [85] for application and resource mapping [52], initial designs of Grid-enabled libraries [61, 29, 62], a prototype system to construct performance models from application binaries [58], and a prototype tool to build mappers and performance models from application task graphs [57].

Finally, to support research and empirical evaluation, we have constructed two research testbeds. The *MacroGrid* consists of Linux clusters with GrADS software installed at several participating GrADS sites. This testbed has been used to validate the prototype framework [96] and for the application studies described above. (Some of the software systems developed for the MacroGrid, such as the VO-grid information system [83], have been incorporated into the NSF Middleware Initiative (NMI) and other standard Grid middleware.) The *MicroGrid* is a software testbed that runs on clusters and permits experimentation with extreme variations in network traffic and loads on compute nodes [81]. This software, which will make it possible to validate the rescheduling function of GrADSoft, will have an initial public release in May 2003.

GrADS also spawned a significant body of work [88, 87, 11, 22, 38, 47, 50, 99, 86, 98, 77, 5] not discussed here. The GrADS web site (`http://hipersoft.cs.rice.edu/grads/`) contains preprints of most of these papers.

### D.2 Proposed New Research

As many have noted, the long-term success of any technology rests on its ultimate invisibility. New technologies are successful when they empower and enable new human activities rather than enmeshing their users in the details of the technology. Desktop software packages and the web empowered millions of new computer users, few of whom understood or even desired to understand how the underlying systems work. The Grid, in its current nascent state, bears many similarities to earlier technologies—the potential is great, but its impenetrable complexity limits its use to only a small cadre of sophisticated users and software developers.

Despite these usability limitations, the Grid's enormous potential has catalyzed large investments in Grid technology and infrastructure by both industry and government. These investments have focused on the development of low-level, interoperable protocols, services, and software packaging, much as early Internet development did. The result is some degree of interoperability and technology invisibility. However, the

critical issues that VGrADS will address (e.g. simplified Grid application development, user-friendly services, and resilient execution contexts on the changing Grid) have been the subject of only limited research.

These observations motivated GrADS, which explored and validated the power of intelligent resource selection services, automatic application scheduling and adaptation, and quantitative performance contracts to track and control application behavior. While this research has generated important new results, it also exposed the key challenge for VGrADS—the explosion of complexity in the Grid resource environment makes timely collection of resource status and rapid, intelligent choices for adaptation by compilers and schedulers difficult.

VGrADS will simplify application and Grid service development by introducing a powerful set of appropriate abstractions and service constraints. These abstractions will constrain the execution context from above with performance contracts and from below with virtualized resource capabilities. In VGrADS, we will explore this approach using resource abstraction classes, vgrids, and program preparation schemes for vgrids. We will collaborate with discipline scientists on a spectrum of Grid applications to drive research, development, and testing of the VGrADS software infrastructure. Figure 1 depicts this vision, with the Grid represented by a layered collection of abstractions and associated service provisions.

At the lowest level of Figure 1, physical resources form building blocks for resource classes, each with an associated set of well-defined characteristics. As the Grid grows to include millions of heterogeneous systems, identifying resources with appropriate attributes for an application (e.g. a 32-node x86 Linux cluster with 4 GB node memories) becomes increasingly problematic. Hence, abstracting irrelevant detail and simplifying resource identification are critical to Grid scaling. This is particularly true for dynamically selecting resources in adaptive VGrADS applications.

The vgrid layer is the heart of Figure 1, providing hierarchical views of distributed resource collections, their properties, and their structure. Above the vgrid layer, programming tools and abstractions target simplified (and perhaps customized) vgrid resource views. Below the vgrid layer, resource managers, schedulers, and run time systems condition and ensure expected behaviors (or ranges of behaviors) for resource and service collections and classes. Thus, a vgrid provides programming tools and applications with timely and scalable access to both resource information and resources. As a result, vgrids provide a simpler information target for resource monitoring, application performance monitoring, and performance prediction.

The multi-level abstraction of Figure 1 is consistent with current Grid research trends toward higher-level virtualized services (e.g. reliable storage) [33]. It realizes the principle of invisibility described earlier, enables separation of concerns, eases the burden of Grid programming and increases the number of resources that can be incorporated effectively. The research challenge lies in determining appropriate abstractions, resource definitions, policy implementations, and programming interfaces.

The VGrADS project will focus on understanding appropriate vgrid abstractions for application use, and the types of virtualization (i.e. specifications and abstraction schemes) that most benefit application development. Our approach will be experimental, resting on two principles: (1) deep engagement with domain scientists who are developing large-scale Grid applications and (2) iterative specification, development, testing, and experimental assessment of the vgrid infrastructure. Figure 3 illustrates this process, which is similar to that used in the successful GriPhyN project. Research is guided by the challenges faced by domain scientists in multiple areas (see § D.2.3), each chosen to expose a unique set of Grid problems. Application work highlights aspects of the evolving system that should be the subjects of additional research. The MicroGrid and MacroGrid testbeds, together with new VGrADS software, will form the basis for rigorous experimentation with the applications. This process will produce software prototypes and infrastructure suitable for use by the scientific community.

An iterative cycle of research, prototype development and experimental assessment with large-scale applications is central to our research plan. As Table 1 in § D.4 shows, during the five year project lifetime, we plan to complete two major cycles of research and testing, culminating in a public software release for each cycle. In the remainder of this section, we describe the VGrADS execution system, program preparation system, and initial application suite.

### D.2.1 VGrADS Execution System (Chien, Berman, Casanova, Dongarra, Kesselman, Reed, Wolski)

The VGrADS execution system, as shown in Figure 1, consists of two key elements: layered resource abstractions (vgrids, resource abstraction classes, and individual resources) and execution system technologies that provide critical capabilities atop those abstractions (scheduling, on-line performance monitoring and fault

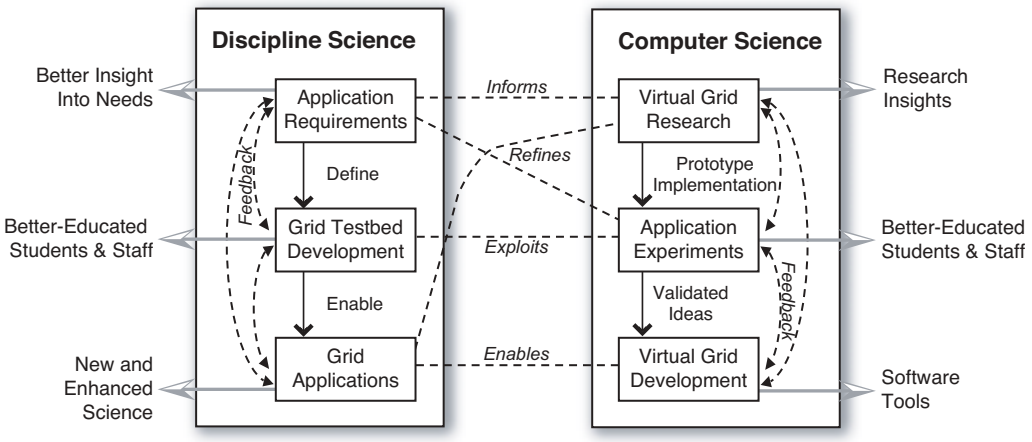


Figure 3: VGrADS Collaborative Engagement.

tolerance). While we discuss these separately to highlight key technical differences, they will interoperate within a uniform services model (such as OGSA [33]) to form a comprehensive execution environment.

**Resource Virtualization.** VGrADS resource virtualization (with corresponding resource quality and performance guarantees) enables separation of concerns, easing the burden of Grid programming and simplifying resource inclusion [83]. The system we propose employs a two-layer approach to resource virtualization. First, resource inspection, characterization, conditioning, and classification are combined, enabling categorization of available resources into *resource abstraction classes*. Resources within the same class are deemed equivalent with respect to the categorizing attributes, which may be static (e.g. operating system type or machine architecture), dynamic (e.g. forecast available processing speed or I/O performance), or both.

Second, application, user, resource, and site administration data and policies are used to select resources from various classes and assemble them to form vgrids that meet specific application needs. Performance contracts—a powerful abstraction developed as part of GrADS—specify the topologies for vgrid structures and provide a mechanism for monitoring their stability. Intuitively, resource abstraction classes characterize the available "raw material," and vgrids impose a structure over that material as mandated by application and user requirements.

***Virtual Grids.*** A vgrid is a collection of resource aggregates that is considered when the scheduler instantiates an application's specific *abstract parallel machine* or *abstract component machine* (see §D.2.2). Vgrids can be constructed for individual users, applications, or virtual organizations. Vgrids are a scoping and organizing mechanism to manage the tremendous resource complexity of emerging Grid environments.

More specifically, vgrids structure the sets of resources most relevant to an application, and they provide the needed behavioral guarantees for resource ensembles and their interactions. For example, if an application needs a specified amount of physical memory, a vgrid may aggregate that capability from the resources belonging to different resource abstraction classes, each specifying resources having a different memory capacity and reliability. Individual resource characteristics are embodied in the classifications, allowing the vgrid to compose these characteristics to meet the application needs.

Vgrid specifications should include resource requirements (e.g. reliability, tightly-coupled performance or large memory) user policy requirements (e.g. access permissions, priority or storage quota), compatibility with historical user and application usage patterns, and proximity to critical data collections. Clearly, there are competing tensions between resource breadth (inclusiveness) and efficient selection (exclusivity). Consequently, vgrids will contain rich hierarchical structure, enabling program preparation tools, run-time schedulers, and applications to work at different levels of abstraction, balancing rapid decision making against fine-grained control when selecting resources.

For example, a data-intensive application might require a large, central data node and a set of well-connected compute nodes (i.e. a master/slave organization). The vgrid for such an application would provide a set of "aggregates" each of which is a data node (with replicas of the desired data) well-connected to compute resources, where the compute sites are not necessarily well-connected to each other (i.e. in a "star" topology). Such a vgrid dramatically simplifies resource selection for the application.

***Resource Abstraction Classes.*** Resource abstraction classes categorize the raw capabilities of physical resources while eliding the unmanageable detail arising from the wide diversity that organizing them as a Grid brings. For example, there are many thousands of hardware configurations in even a small slice of the Internet [76]. Without abstractions, this explosive diversity will require application developers and run-time systems to identify and track physical resource locations, attributes and behaviors, system software versioning, etc.—a daunting task.

As part of VGrADS, we will develop scalable techniques for dynamically creating and maintaining *resource abstraction classes*. By providing a hierarchical approach to managing the Grid, these abstraction classes will reduce the complexity of resource discovery and acquisition, as they will pool resources into logical collections from which useful aggregates, with known properties, may be built.

Resource abstraction classes can be formed based on *simple static properties* such as software or hardware configuration, physical connectivity, ownership and location, more *complex static properties* such as cluster structure and batch scheduling capabilities, or *dynamic properties* such as availability, deliverable compute performance or deliverable communication performance. These characteristics can be derived by inspection, statistical characterization, prediction (e.g. NWS-like forecasting techniques [97, 95]), or by declaration from an appropriate, trusted authority. These properties can also be negotiated, arising from service-level agreements and Grid protocols such as SNAP [23]. In some cases, effective resource classification may leverage extant software virtualization technologies. Thus, to support resource abstraction classes we will explore use of commercially available virtual machines (e.g. VMWare, J2EE, and Connectix) to deliver performance as a commodity and reduce the overhead of software configuration management for Grid resources. We will also investigate mechanisms to ensure performance stability guarantees that are most appropriate for constructing resource abstraction classes. As such, our work will balance optimized local system efficiency and overall Grid resource stability—a tradeoff unexplored by previous research.

**Execution System Services.** In addition to vgrid resource abstractions (and underlying Grid information services, security, and resource access mechanisms), the VGrADS execution system must provide system services to support COPs executing on an abstract parallel or component machine, while meeting the expectations of a performance contract using the underlying vgrid resources. Key services include on-line performance monitoring, fault tolerance, and scheduling for vgrids. Scheduling services enable rapid, efficient resource selection within a vgrid; fault tolerance services enhance the reliability of a Grid resource; and performance monitoring services enable application adaptation based on surprising realized performance.

***On-line Performance Provisioning.*** Our GrADS experiments show that understanding Grid performance requires on-line assessment of multilevel interactions among applications and changing Grid resources. Given this assessment, dynamic adaptation is often necessary to deliver soft performance "guarantees" to Grid applications. The VGrADs framework provides simplifying abstractions for multilevel performance inspection, exporting critical monitoring information for use by programming tools.

Within this framework, we will develop novel techniques for *on-line performance evaluation* and adaptive multilevel assessment for physical resources, resource abstraction classes, and vgrids. By evaluating these models in terms of their published accuracy, the scheduler can select appropriate vgrids for application execution (e.g. based on a balance of performance and reliability) and validate run-time behavior by comparing observed and model-predicted behavior. Additionally, these models will enable post-mortem analysis for construction of new vgrids.

Our new models will combine work on application signatures (compact representations of trajectories within behavioral metric spaces) [53] and performance estimates for time to completion. We will explore statistical techniques that enable scalable performance and reliability estimation for vgrids containing tens or hundreds of thousands of entities, while explicitly balancing measurement overhead and specificity. Given performance models and intelligent measurement, on-line evaluation and remediation will trigger multilevel adaptation based on an extended notion of the original GrADS performance contract. We will build on non-parametric adaptive statistical techniques for reasoning about time varying behavior, fuzzy logic for decision making, genetic programming techniques for evolving new control systems, and market economics for decision making.

***Reliability Provisioning.*** The large-scale Grid differs from parallel systems in having greater behavioral uncertainty, both in expected performance and reliability. Just as performance provisioning reduces application complexity by associating software performance guarantees with the underlying vgrid, reliability

provisioning enables long-running Grid applications atop intrinsically unreliable components.

To provide reliable vgrids, we will explore techniques for fault tolerance and computation migration. Fault tolerance techniques can help exploit the computational resources of the Grid without compromising its highly flexible design. Similarly, vgrids hide the identity of physical resources, enabling migration of application components while preserving the vgrid.

We will develop reliability scaling models based on measured behavior of Grid components. To enable resource classification on the basis of reliability, we will exploit our measurement infrastructure, based on both active probing of Grid resources and passive measurement [60] as well as commercial packages to populate a database of failure intervals and modes. This work will also use measures of system failure and recovery modes (e.g. software memory and recoverable disk errors, temperature sensors, and network errors), to develop predictive techniques for failure modes.

Using this failure database, we will develop scaling models that estimate component and aggregate availability. The resulting scaling models will enable construction of vgrids which meet a variety of reliability specifications. These vgrids can then be combined with fault-tolerant MPI libraries, in-memory checkpointing and recovery, and task migration to meet a desired level of application reliability. Intuitively, this approach shifts computations to new resources when a current resource fails or cannot meet performance guarantees.

***Virtualized Scheduling and Resource Selection.*** Effective resource selection and scheduling (of both data and computation) are critical performance factors for Grid applications that demand high application performance with maximum resource efficiency, and that acquire resources dynamically. Moreover, Grid resource selection mechanisms must be fast and robust because they can dramatically increase the application's run-time overhead.

Current Grid systems have limited ability to make appropriate resource selection, mapping, and scheduling decisions, adopting *ad hoc* heuristics that are limited to simple applications and do not scale to large, complex Grid resource environments. Vgrids will provide flexible abstractions and hierarchical structure that let the VGrADS schedulers focus on the available computational resources to make rapid, and intelligent resource selections in large, complex Grid resource environments.

A VGrADS application identifies its resource requirements and selects a vgrid constructed from suitable resources (in terms of desired performance, reliability, etc.). The application manager (which contains the scheduler) then selects appropriate resources from the vgrid (based on current performance data and available predictions and guarantees) and negotiates with resource managers to gain access to these resources. Early GrADS experience indicates that resource virtualization is critical to enable better scheduling decisions [26, 64, 83]. We will build on these results, developing virtualized scheduling and resource selection strategies that accommodate a range of application-centric performance notions (e.g. turnaround time, throughput or reliability), resources with multiple usage modes and policies (e.g. sharing model, priorities and reservation systems), and optimization of system utilization for local efficiency metrics.

***Grid Economies and Policy.*** In addition to this application-centric approach to resource selection and scheduling, Grid resource management must also balance application demands against excessive resource consumption, which can lead to congestive system collapse. Resource classification is critical to decentralized scheduling, allowing resources to be treated as commodities, enabling a scalable approach to dynamically-priced, decentralized resource economies. Our recent work indicates that economic-based scheduling has provable stability qualities that are critical to meeting these challenges [99, 98]. We will explore novel selection and allocation policies, based on economic and social science models, that promote scalable, stable, and performance-efficient scheduling. Building on this work, we will develop *VGrADSBank*, a scalable distributed resource pricing exchange. VGrADSBank will use resource abstraction classes as a baseline abstraction and will be integrated as an extensible part of the Grid information system to support a variety of approaches to economic-based scheduling.

### D.2.2 Programming Tools (Kennedy, Cooper, Dongarra, Johnsson, Koelbel, Torczon)

Programming tools research will seek to raise the level of abstraction at which the programmer sees the Grid, thereby reducing the difficulty and increasing the accessibility of Grid application development. Initially, we will explore two models as targets for programming: the *abstract parallel machine* and the *abstract component machine*. A critical challenge will be to map these models efficiently to the abstractions provided by the VGrADS execution system. This work will build on the strategies for constructing mappers and

performance models that we developed for the current GrADSoft prototype.

**Abstract Parallel Machine.** The *abstract parallel machine* model treats computations as collections of parallel communicating tasks without any consideration of the mapping to physical hardware or load balancing. Each task can run on any computational element, and each task can communicate with any other (i.e. full communication connectivity). Intuitively, the abstract parallel machine idealizes actual parallel and distributed computer systems by letting the programmer ignore the performance of the computational and communication resources.

The programming tools must map this idealized model onto a vgrid formulated by the execution system (§D.2.1). The key challenge is automating the process of *load matching*—mapping the computational components onto resources in a way that minimizes some objective function (e.g. time to solution or cost). Successful load matching will require the execution system to expose the abstract performance structure to the programming tools in a way that hides unnecessary resource detail, while providing critical data via abstraction interfaces. Because these interfaces are the essential mechanism for providing performance and reliability estimates for computation, communication and memory availability, their design must optimize the trade-off between detail and abstraction.

The GrADS project (indicated in gray in Figure 1) explored a specific instance of this problem—mapping applications written for an abstract parallel machine to a specific instance of a vgrid (a reliable x86, Unix, MPI environment; characterized by compute speed, memory availability, and with complete communication connectivity [83]). VGrADS will build on these results and on experience with the MEAD application (see § D.2.3), to formalize, extend, and refine this approach. The work will include research on improved tools for constructing executables, mapping executables and computations onto a vgrid instantiated by the execution system, and moderating execution to improve performance and reliability.

***Mapping Applications to Virtual Grids.*** In GrADS, the basic strategy for mapping both task and data parallelism was to model a parallel program as a task graph, which was then aggregated onto real compute resources using graph clustering [51, 8, 46]. A fundamental new challenge in VGrADS is to develop mapping techniques that effectively exploit the virtualization strategies in the execution system. To do this, the mapper must construct a *virtualized task graph* whose computations are grouped into *granules*—small tasks that represent the minimal amount of work suitable for execution on a single vgrid node.

The VGrADS mapping function will be split across two phases: *mapper construction* and *mapper execution*. Mapper construction consists of all those activities that can be performed statically through program analysis, without knowledge of the available Grid resources. Mapper execution occurs at program launch time or at run-time (in response to a performance contract violation).

The constructor must first examine the application to identify and parameterize the space of vgrids that are acceptable for execution. Next, it will refactor computations to control granule size. In effect, this virtualizes the tasks, creating a virtualized task graph. Granules should be large enough to maintain a good ratio of computation to communication, or be composable so that grain-size can be determined dynamically. This strategy simplifies the scheduler's task by letting it assume a roughly uniform granule size. Choosing sizes effectively is a key research challenge.

When the mapper executes, it aggregates granules into groups to run on specific resources within the instantiated vgrid. It must use performance models of the application and performance forecasts for the instantiated vgrid to assess the expected application performance. Knowledge from these models and forecasts will let the mapper make intelligent decisions on how to aggregate granules and where to execute them in the vgrid chosen by the scheduler.

The interaction between mapping strategy and scheduling and between performance modeling and forecasting highlights the need for an integrated approach to the VGrADS suite of problems. It also shows how critical the abstractions and interfaces are to the capabilities of the resulting system.

***Performance Modeling.*** Both mapping and scheduling of an abstract parallel programming need performance models to evaluate the trade-offs between different maps and schedules. The construction of accurate performance models is a daunting challenge. In VGrADS, we will exploit prior experience in the GrADS project along with ideas generated in other systems, such as POEMS [2], to build performance models as required. This work will exploit both the on-line performance monitoring capabilities from vgrid services and application-specific performance models.

The GrADS system builds hybrid performance models by combining static estimates, measurements of

actual execution, and dynamic forecasts of resource performance. It uses tuning runs on an instrumented executable to determine the performance sensitivity of computational kernels to varying data sizes [58]. Using this data, it constructs an executable model that replaces execution of kernels with estimates, determines the scaling effects, and replaces communication with estimates of the communication time. The resulting models were sufficiently accurate to use in mapping tasks to uniform computational nodes.

To extend this to heterogeneous vgrids and to model the effects of granule aggregation, we will need to involve both the mapper and resource performance forecasts from the execution system. Single granule models can be built as in GrADS. To construct global performance models, the VGrADS tools will invoke the mapper, using cost-of-communications estimates from the instantiated vgrid. The mapper will cluster granules onto specific resources. At this point, the tools can construct and run an integrated performance model for the mapping to obtain a more precise estimate of execution time. Once again, the interplay between application-specific performance models and vgrid-specific performance predictions makes the modeling task a challenging research problem.

***Moderating Execution Performance.*** The VGrADS toolset will also include components that aim to improve the performance of an executing configurable object program (COP) and moderate its performance variability. This work will be applied in the compilation tools that construct COPs and in the binder and mapper at run time.

At COP-construction time, the tools will use automatic strategies to pre-tune codes to vgrid parameters well in advance of execution. To this end, they will apply the ideas of telescoping languages (described more fully under "Library Preparation and Installation") to identify opportunities for optimizing sequences of operations. In addition, they will build on strategies pioneered in CMSSL [44, 45], FFTW [35, 36], UHFFT [56, 61, 62], SPIRAL [66] and ATLAS [92] for tailoring code to the specific machines on which an application will run, without the overhead of launch-time optimization.

Immediately before execution, the binder has a final opportunity to further tailor the code for a specific computational node. We will use a virtual machine strategy to extend COP execution to new hardware architectures, coupling fast emulation with bind-time and run-time tailoring to achieve acceptable performance. We will also explore hybrid mapping strategies that ship object code to the data when doing so reduces communication volume.

**Abstract Component Machine.** Hiding the details of parallelism from the user is one way to provide a higher level of abstraction. We will explore a strategy that embeds parallelism within professionally-developed software components that target either an abstract parallel machine or a vgrid directly. These components could be realized either as Grid services or as constituent application code that is pre-staged onto some collection of compute resources. Applications constructed in this way can be seen as programs for an *abstract component machine* whose atomic operations invoke these components. For example, it should be possible to build an application as a MATLAB program that invokes Grid-deployed numerical solver components. We will use the abstract component machine model as the basis for two of our application collaborations described in § D.2.3: Bioimaging (GEMS) and the Encyclopedia of Life (EOL).

As a part of this effort, we will explore the construction of problem-solving systems based on component integration frameworks for Grid execution. As a by-product, we will produce a prototype component integration framework that supports the development of applications via scripting languages that invoke high-level domain-specific components. The Common Component Architecture (CCA) [7] is an excellent starting point for our specification. To achieve reasonable performance, however, these applications will require careful mapping of components onto the chosen vgrid along with extensive cross-component optimization (probably in the binder). This strategy should enable new levels of scalability for applications such as GEMS and EOL while eliminating the need for expert system-level programmers.

***Workflow Model.*** Many of the strategies for component machine implementation build on previous Grid projects focused on workflow-style applications and on our previous work in the GrADS project. In projects like GriPhyN [39], workflow is represented as a directed acyclic graph (DAG) in which the components to be used are nodes and the edges represent dependencies due to data transmission.

This DAG-based workflow approach is quite effective for certain applications, but it hides significant opportunities for improvement. Task graph clustering strategies like those described for the VGrADS abstract parallel machine might group pairs of tasks onto the same virtual resource, eliminating the opportunity to pipeline the execution as a way of exploiting overlap. To fit a pipelining strategy into the mapper, the

constructor should consider it as an alternative to whole-file transfer between components.

In cases where the program execution structure depends heavily on the input, the VGrADS scheduling mechanisms will use dynamic scheduling and rescheduling. For example, the GrADSAT application (§ D.1) requires spawning and scheduling of new tasks at run-time. This data-dependent execution profile presents an important challenge — the mapper must group tasks and granules to a granularity that improves performance without precluding the advantages of dynamic dispatch. Again, choosing the right granularity is critical. It requires the appropriate interfaces to share information between the execution system and the program preparation system. Our work on the GEMS application will employ the workflow model (§ D.2.3).

***Component Services Model.*** To implement the abstract component machine, the binder must be able to link and tailor executables efficiently. In § D.1 we discussed how the GrADS binder prepares the COP for execution, including tailoring code to the resources on which it will execute. In VGrADS, the binder must accommodate a grid services model where some software components and data archives are pre-installed on the Grid. This "just-in-time" service model will complicate both binding and program launch. However, it has the advantage of encouraging late-stage optimization [22] and the use of ATLAS-style tuning [92] of individual components to the resources on which they are installed.

A key challenge of this effort is to integrate a COP using information (mappers and performance estimators) specific to the components but in a way that exploits the scalability of vgrids. Under the services model, the VGrADS execution system will not only indicate which Grid resources can execute a specified set of software components but also provide a performance model for each component on each resource. These attributes must be transparent across the vgrid layer so that the mapper can invoke a global constructor to combine them into a performance model for the entire application. This process will also involve integration of performance models for virtualized resources provided by the VGrADS execution system. In some cases, the integrated performance model itself may need to run on the Grid.

***Library Preparation and Installation.*** To support the component services model, VGrADS needs mechanisms for installing component libraries on specific Grid computational resources and registering them with Grid information services. It must be possible for the launch-time mapper to evaluate and exploit combinations of components that can be optimized to run together on that resource with higher performance than if they are used individually. To accomplish this, we plan to build on the telescoping languages strategy [47, 59], which speculatively optimizes sequences of component invocations when the component library is created. The goal is to produce highly-optimized library services for specialized contexts that are likely to occur frequently in programs using the library.

For VGrADS, this optimization would occur when the library is installed on a particular platform. At that time, knowledge of the target platform allows intense optimizations such as ATLAS-style performance tuning [92] as well as more standard transformations [22]. The optimizing installation process will be computationally intensive, but it only needs to be done when new libraries are installed—perhaps a few times per year. After installation, the tools must register the target machine with the VGrADS information system, to indicate that it offers the library and provides performance models and mappers for the components. This lets the vgrid layer serve this information to the abstract component machine mapper.

### D.2.3 Application Partnerships

The potential applications of Grid technology span the entire continuum of computational science and engineering. Hence, we believe it is critically important to choose a set of "driving applications" that represent distinct, representative points along the continuum. The three target applications, drawn from biology and engineering, all contain a significant number of interacting computation and data management components, and they typify the Grid needs of the next generation of computational science applications.

These driving applications will guide both VGrADS research and software development, yielding a flexible, robust vgrid infrastructure. Equally important, the domain scientists associated with each application are committed to deep and substantive collaborations with the VGrADS team.

As previously noted, our research and development model, illustrated in Figure 3, is an iterative one. Insights from experiments will shape and inform successive generations of VGrADS software. Based on experiences and deployment, we will add new applications during years 4 and 5 of the effort. For this use, we will choose scientifically important applications that can drive VGrADS development into new areas.

**Grid Enabled Molecular Structure determination (GEMS).** (Lead Scientist: Wah Chiu (Baylor); VGrADS Lead: Lennart Johnsson (Houston)) The determination of 3-D structure of large macromolecular complexes ("particles") made up of multiple molecular components is an important goal in structural genomics and proteomics. Electron cryomicroscopy that has been successfully applied to virus pathogens [14, 102, 101, 43, 21] also has the potential to reveal folds of individual molecular components and their interactions responsible for the normal and abnormal states of the complex in proteomics research [34, 75]. Biochemical and electron cryomicroscopy improvements in combination with the Grid hold the promise for macromolecular structure determination to become a high throughput and high accuracy technology for structure determination, data mining, sharing and visualization in proteomics research.

In electron cryomicroscopy each microscope can produce several terabytes of data a year and many groups have multiple microscopes. The frames from the microscopes are deposited in a database for access by a suite of software tools designed to perform reconstruction, mining, and visualization of 3-D structures. A project may require thousands or tens of thousands of frames with dozens to thousands of particles per frame. Depending upon the object, structure determination may need tens of thousands to millions of particle images. The computational requirements for this phase alone range from petaflops to exaflops.

The processing steps for structure determination are: localization of particle images within frames, power spectrum characterization on a per-frame basis, initial structure determination, and a complicated iterative refinement process. Multiple iterations of image deconvolution and structure refinement are necessary, with the number of cycles depending on the structure and desired resolution. Some of the stages have modest parallelism and computational requirements while others are highly parallel and require extensive computation. Furthermore, some stages operate on frames, others on particles or collections of particles. This poses challenges for developing the VGrADS resource abstractions, vgrids, component machine models, and parallel machine models. The VGrADS resource abstractions, together with the mapping and launch mechanisms, will facilitate the use of the Grid for GEMS and similar applications with diverse computational and data resource needs. The database represents a fixed Grid resource, whereas the various computational resources can be selected to optimize use of available Grid resources while meeting component needs.

Initially, we will develop and evaluate VGrADS resource abstractions by creating COPs for the structure determination package EMAN [54, 55] developed at Dr. Wah Chiu's National Center for Macromolecular Imaging. EMAN is designed to automate as much of the single particle reconstruction process as possible and contains a number of tools for post-processing the reconstructed volumes [42].

**Encyclopedia of Life (EOL).** (Lead Scientist: Mark Miller (SDSC); VGrADS Lead: Fran Berman (SDSC)) EOL [30] is an ambitious collaboration led by the San Diego Supercomputer Center to analyze and archive the three-dimensional structure and function of protein sequences from all publicly available genomes. EOL will have dramatic impact on fundamental and pharmaceutical discovery research as it integrates all information about proteins across the biological scales. There is a natural synergy between EOL and GEMS, as results obtained in GEMS calculations can be stored in EOL's federated data archive.

The main computational EOL activity is the characterization and functional description of protein sequences from all publicly accessible genomes—currently over 800 genomes. Using current algorithms and technology, this calculation will require approximately 800 processor years. The number of raw DNA sequences is doubling every 6-8 months. Hence, leveraging available Grid resources is critical to EOL's success.

EOL sequence characterization is a loosely coupled application—each genome and each sequence within a genome can be characterized independently. These computations are often I/O intensive and require access to biological sequence databases (which must be distributed, replicated, and updated periodically), thus requiring data-aware scheduling strategies. More importantly, each sequence characterization involves 5 to 10 different steps, where each step is potentially a data parallel application (i.e. a parallel work flow). Such mixed parallelism is known to pose difficult problems [17, 12, 31, 91, 82], and will be an excellent test for VGrADS technology. Due to mixed parallelism, EOL requires both VGrADS machine models: the abstract component machine model to describe the application at a high level, and the abstract parallel machine model at a low level. We will apply the VGrADS programming tools and methodology to EOL to obtain a version of the application that is amenable to adaptive execution on the Grid.

From an execution perspective, mixed parallelism raises challenges in terms of application scheduling, and several heuristics have been proposed in the literature [67, 68, 70, 71, 15, 9, 10]. Due to the nature of Grid environments (e.g. dynamic performance characteristics, interconnection over complex networks) it is extremely difficult to apply these techniques directly to EOL. The vgrid concept provides both the level of

abstraction and the hierarchical structure necessary to reason about resource selection. We will leverage previous scheduling research results to develop new scheduling strategies targeted to vgrids.

**Modeling Environment for Atmospheric Discovery (MEAD).** (Lead Scientist: Brian Jewett (UIUC); VGrADS Lead: Dan Reed (NCSA)) MEAD [94, 93] is a scalable research framework for exploring ensemble prediction and conducting parameter studies to better understand the behavior of severe weather phenomena (e.g. heavy rain producing mesoscale convective systems, damaging supercell/tornadic storms and destructive hurricanes [69]). In both cases, one must conduct a large suite of simulations and compare the results. These comparisons may be used to provide probability forecasts and estimates of forecast skill, to determine the sensitivity of severe weather structure and evolution to environmental conditions, and to understand the dynamic physical processes that lead to the development and decay of weather phenomena.

The number of simulations in a suite has typically been constrained by the computational and labor-intensive nature of the process. Computational needs for any particular simulation in a research environment are within the capabilities of a mid-sized system. However, when hundreds of simulations are needed for a study (e.g., such as one being undertaken to study potential tornado development resulting from a positive interaction of individual storm cells [41, 40]), the computational requirements become 100 to 1000 times greater. Further, the analysis, data mining, and visualization support needed to glean insights from the resulting tens of terabytes of data present a significant data management challenge, one for which current software tools are ill-suited.

With the advent of the Grid and the NSF-funded TeraGrid, it is now feasible to conduct these simulation suites and subsequently analyze, mine, and visualize them, as well as store the results of all components at distributed sites. For example, coupled atmospheric and ocean models can be executed on different platforms and can be linked effectively across the Grid. The goal of the Alliance MEAD expedition is the development and adaptation of the cyberinfrastructure for this task. Portal Grid and web infrastructure will enable launching of hundreds of individual Weather Research and Forecasting (WRF), Regional Ocean Modeling System (ROMS), or coupled WRF/ROMS simulations on the Grid, in either ensemble or parameter mode. Discovery and use metadata coupled to the resulting terabytes of data will then be made available for further study and for educational purposes. A researcher using the MEAD work flow environment will be able to configure and integrate model simulations, manage resulting model and derived data, and analyze, mine, and visualize this data using a collection of software that can run on a variety of Grid platforms.

MEAD provides a third set of problems for Grid execution – ensemble scheduling, time to completion estimation, and resource management. In conjunction with MEAD, VGrADS will not only enable suites with large numbers of simulations to be carried out but also enable suites of suites for studying the vast array of weather scenarios that are possible in nature.

### D.3 Broader Impacts

The PIs are firmly committed to integrating education, human resource development, and technology transfer programs into VGrADS. In education and human resources, we will extend the successful Education, Outreach, and Training (EOT) programs of our previous center-style collaborations (i.e., CRPC, the Alliance and NPACI) to introduce Grid computing concepts. These EOT efforts will be led by Richard Tapia and Cynthia Lanius at Rice University, with participation from personnel at all sites. In technology transfer we will extend our contacts with academia, government, and industry to highlight VGrADS technology and distribute software and ideas. Although the distribution efforts will focus on the VGrADS software repository at Rice University, all sites will contribute (e.g., via participation in the National Middleware Initiative).

#### D.3.1 Education, Outreach, and Training (Tapia, Berman, Cooper, Kennedy, Koelbel, Reed, Wolski)

In the VGrADS effort, we will pursue projects that address graduate, undergraduate and pre-college education, with special attention to the needs of underrepresented groups.

Richard Tapia, who has received several national awards for his work in this area, has designed the VGrADS EOT program to focus on supporting Minority Students in Majority Institutions (MSMI). These students are a substantial pool of talent available to expand the ranks of science and technology workers. However, due to their backgrounds they are often not encouraged to pursue the careers they are capable of. For example, the valedictorian of a weak urban high school who is accepted into a top-tier college most likely will start far behind her classmates academically. Yet she may have the talent to succeed if only given

adequate encouragement and help in catching up. One way that we will encourage our MSMI students is by aggressively supporting two key meetings devoted to increasing diversity in computer and computational science. The Grace Hopper Celebration of Women in Computing (`http://www.gracehopper.org/`) is a well-regarded series of conferences designed to bring the research and career interests of women in computing to the forefront. The Richard Tapia Celebration of Diversity in Computing (`http://www.ncsa.uiuc.edu/Conferences/Tapia2003/`) is a similar conference series dedicated to the growth of diversity in computing and related disciplines. Several VGrADS PIs have been associated with both conferences, including Fran Berman and Ken Kennedy as invited speakers. For both conferences, we will make special efforts to submit appropriate tutorials, panels, and papers based on the VGrADS research. More concretely, VGrADS will fund the travel of appropriate graduate and undergraduate students to the conferences to enhance their knowledge, reputations, and contacts. MSMI programs like the one we propose support these talented students, directly attacking the attrition of good people from the science and engineering workforce.

At the pre-college level, VGrADS personnel will provide content to existing programs. In particular, Cynthia Lanius, Richard Tapia, and Keith Cooper are the PIs of the Computer Science Computer and Mentoring Partnership (CS-CAMP) project (`http://ceee.rice.edu/cs-camp/index.html`). This project seeks to enhance the interest and persistence of female students in pre-college computer science. The Rice PIs are committed to making presentations at CS-CAMP sessions for teachers and high-school girls and helping provide curriculum materials. As they become available, the EOT/PACI network will be used to disseminate these materials nationally. In addition, SDSC has its own education and outreach programs, and will disseminate and integrate the VGrADS materials into its existing programs. NCSA will ask VGrADS personnel to participate in other NSF supported workshops, including the National Computational Science Institute (`http://www.computationalscience.net/`), to share the tools and resources with faculty through regional summer workshops and the annual SC conference.

At the undergraduate level, VGrADS activities will bring Grid computing into the classroom through new courses, involve students directly in the research programs of the PIs, and build support communities for student participants to improve retention of those students in technical degree programs. Different activities will target non-technical students and computer science and engineering majors.

For the non-technical student, we will focus on general education courses in technology. Ken Kennedy is developing a course, *Information Technology Architectures*, that teaches the high-level principles of designing and building information technology systems without extensive programming. This course will be a cornerstone of Rice's general education offerings in computer science. Once this course is tested at Rice, we will work to establish similar courses at other VGrADS sites.

For science and engineering students, we will focus on promoting graduate study among students at our own institutions, targeting women and underrepresented minorities in particular. We will expand Rice University's existing AGEP program (`http://rgs.rice.edu/Grad/agep/`), which is itself an extension of the CRPC's very successful "Spend a Summer with a Scientist" program. This program brings undergraduate students to Rice to participate in summer research with established faculty and other activities to help them in entering scientific careers. We will fund up to four additional AGEP participants who will work on Rice VGrADS research. By aggressively recruiting the students from other VGrADS institutions, we hope to establish "core groups" at those sites to support others there. We also intend to apply for REU supplements which will be used to incorporate students into other programs. For example, SDSC's summer program offers special seminars and short courses for students, in addition to their normal research opportunities. VGrADs REU students located at UCSD will be welcomed and integrated into the SDSC REU program.

At the graduate student level, VGrADS will enhance the usual graduate studies for these students in several ways. Most importantly, we will initiate an active long-term exchange of graduate students among research groups to broaden their backgrounds and improve the research in both the "sending" and "receiving" groups. Also, we will solidify the collaborations arising from these trips by encouraging students to present their work at regular VGrADS participant workshops. The VGrADS project will directly fund over a century of graduate student study. As VGrADS matures, we will develop a standard academic course on Grid architectures and programming systems suitable for both graduate and undergraduate students. A model for this course will be Wolski's current Grid Computing seminar, in which the students last year wrote adaptive Grid programs using the GrADS-developed VO-Grids abstraction and Grid Economies prototype. Eventually, we expect that this course will be offered at each of the participating VGrADS institutions.

### D.3.2 Technology Transfer (All PIs)

VGrADS will tie its technology transfer activities to its software development cycle. Sites already collaborate in developing software, sharing the code base through CVS. We also maintain an active set of public web pages with current project status and publications. VGrADS will extend this system to make public releases of the software available through web browsers and FTP servers. Ensuring the quality and consistency of these releases and their documentation will be the primary responsibility of the software development team. As explained in §D.4, this team consists of two programmers (at Rice) over and above the research programmers needed to make progress on the experimental software and applications projects in VGrADS. After a public release, access points will be widely advertised by references in our publications, announcements to appropriate newsgroups and mail lists, and tutorials.

Beyond this, VGrADS researchers will work to incorporate appropriate pieces of its software into widely-used toolkits as our system matures. Because most VGrADS researchers are also members of national and international Grid collaborations, each focused on building and deploying production Grid infrastructure, there are myriad opportunities for technology transfer. For example, the NSF Middleware Initiative, which involves Kesselman, Wolski and researchers from the Alliance and NPACI, provides a vehicle for software hardening, documentation, packaging, and transfer. Similarly, Reed is PI for NEESgrid, a national collaboratory and Grid infrastructure for earthquake engineering research, where Kesselman is the leader of software integration. Finally, Berman and Reed are the PIs for the TeraGrid, which is building and deploying production Grid infrastructure atop nationally distributed supercomputing and data storage systems.

Because simply publishing software and research ideas does not lead to their adoption, we will actively present tutorials and demonstrations of VGrADS technology. The annual SC'XY (formerly Supercomputing) conference will be our primary outlet for this. Every year, we will showcase our latest and greatest work in demos on the SC'XY exhibit floor. Before the VGrADS software is publically distributed, we will submit general tutorials on Grid computing—which the SC'XY conference committees have consistently requested. After public releases of our software, we will augment those offerings with tutorials on using the VGrADS system itself. We will repeat these tutorials at other meetings, including the Richard Tapia Celebration of Diversity in Computing, whose organizers have indicated that they would welcome such presentations. The NCSA EOT Division will work with key VGrADS personnel to offer Access Grid seminars on project activities, tools, resources, and materials with the national community. The EOT staff will also work with these personnel to add their content to NCSA training workshops throughout the year. All of these venues will give us opportunities to find new applications partners and encourage the use of our methods.

Technology transfer will also involve industry. All of the VGrADS personnel have connections with some industrial application users and computer industry vendors who are interested in Grid computation. The NCSA Private Sector Program (`http://www.ncsa.uiuc.edu/About/PSP/`) and SDSC's Science and Technology Outreach Program (`http://www.sdsc.edu/Partners/STO/`) are excellent examples of this. We plan to aggressively pursue these connections to ensure that VGrADS solutions are applicable to (and adopted by) real-world systems. On another level, VGrADS personnel are involved in many standards-setting and advisory bodies that will benefit from their expertise in Grid computation. In particular, Chien serves on the Steering Committee of the Global Grid Forum (GGF, `http://www.gridforum.org/`), arguably the best-known organization for Grid computing today. Kesselman chairs one GGF Working Group, as well as being a principal author of Globus, the most popular Grid toolkit. Dongarra and Koelbel serve on the National Academy of Sciences study of the Future of Supercomputing (`http://cstb.org/project_supercomputing/`), which includes Grid computing in its scope. These affiliations will give us excellent connections for moving VGrADS technology into widespread use.

### D.4 Management Plan

Our strategy for managing the VGrADS effort is modelled after the strategy employed by the Center for Research on Parallel Computation (CRPC)—a successful, geographically-dispersed Science and Technology Center—and by the GrADS project. The strategy is based on an ongoing, cyclic pattern of review, planning, and implementation. This approach, which was reviewed and endorsed by the National Academy of Public Administration, allowed the CRPC to both adapt to changes in personnel and institutions and adopt new research directions while remaining focused on its central mission.

**Management Structure.** VGrADS will include researchers from seven institutions: Rice University; University of California, Santa Barbara; University of California, San Diego; University of Houston; University of Illinois, Urbana-Champaign; University of Southern California, Information Sciences Institute; and University of Tennessee, Knoxville. Rice University will be the lead institution, with Ken Kennedy serving as Director. He will be assisted by an Executive Director, Linda Torczon; a Director for Education, Outreach, and Training, Richard Tapia; an Executive Committee (EC); a Facilities Committee (FC); and an Education, Outreach, and Training Committee (EOTC). Figure 4 depicts the reporting relationships among these individuals and committees; the individuals and committees have the following roles:

- The Director is responsible for oversight, planning, budgetary decisions, and reporting VGrADS activities to the NSF. The Director receives advice and assistance from the individuals and committees shown in Figure 4.
- The Executive Director is responsible for implementing decisions made and policies set by the Director, coordinating regular reports required by NSF, planning site visits requested by NSF, and facilitating communication between the Director and the individuals and committees that advise him.
- The Director for Education, Outreach, & Training is responsible for planning, implementing, and assessing VGrADS education, outreach, and training (EOT) programs. He will be assisted by Cynthia Lanius (Rice), who will run the day-to-day VGrADS EOT operations.
- The Executive Committee (EC), chaired by Ken Kennedy, will convene regularly via teleconferences, e-mail, and physical meetings to plan research, education, outreach, and technology transfer programs; review progress; and approve major budgetary commitments. The EC will consist of Ken Kennedy (chair), Keith Cooper, Charles Koelbel, Richard Tapia, and Linda Torczon (Rice); Rich Wolski (UCSB); Fran Berman, Henri Casanova, and Andrew Chien (UCSD); Lennart Johnsson (UH); Dan Reed (UIUC); Carl Kesselman (USC-ISI); and Jack Dongarra (UTK).
- The Facilities Committee (FC), chaired by Carl Kesselman, will convene as required via teleconferences and e-mail to establish policies for managing the VGrADS testbed and to assist in scheduling large experiments on the VGrADS testbed. The FC will include a representative from each VGrADS site that hosts equipment made available as part of the VGrADS testbed.

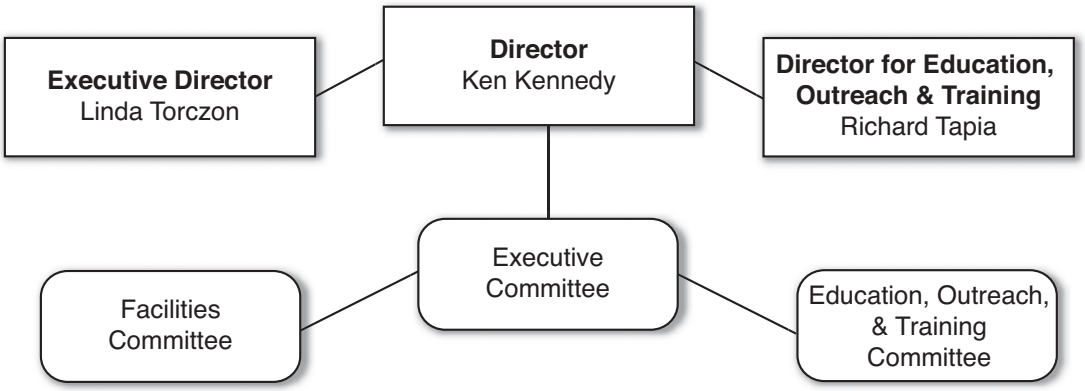


Figure 4: Management Structure

- The Education, Outreach, and Training Committee (EOTC), chaired by Richard Tapia, will convene via teleconferences and e-mail to review and plan VGrADS EOT programs, including efforts to encourage minorities and women to pursue careers in computer science. It will also coordinate EOT efforts among VGrADS sites. The EOTC will include a representative from each VGrADS site.

Each VGrADS site will have both a senior researcher and an administrative contact to assist in operational matters, including reporting and budgeting. Researchers and administrative personnel will communicate regularly through e-mail, the World Wide Web, teleconferences, and physical meetings.

In addition, all sites will have postdoc researchers, research programmers, and graduate students performing the bulk of the technical work. The proposed budget includes 11 technical staff (approximately 8 FTEs) and nearly 20 students, many of whom already work on GrADS. Weekly technical teleconferences and e-mail exchange among VGrADS technical participants will encourage discussion between researchers at the various sites and provide a forum for coordinating VGrADS software development and integration efforts.

The PIs bring to the task a wealth of prior experience in managing large, collaborative, distributed research projects, including the CRPC, the PACI partnerships, and collaboration on the GrADS project.

**Review, Planning, and Implementation.** The VGrADS research and development activities will be organized into three types of activities. The *basic research* activity will draw upon our preliminary research and implementation experiences to develop principles and new research directions. The *infrastructure development* activity will develop the VGrADS software base, incorporating the results of our research, and providing a basis for experimental verification of our ideas and for technology transfer. The *application evaluation* activity will involve joint investigations with application partners associated with one of the VGrADS sites to develop compelling Grid-enabled applications using VGrADS software. The MicroGrid and MacroGrid testbeds from GrADS will be used to test, tune, and monitor these applications.

To coordinate activities among the various projects and sites, VGrADS will hold two internal workshops per year. All VGrADS participants, including students, will be invited. The workshops will serve as forums to review what has been accomplished, refine the overall design of the system based on what has been learned, and assess new ideas and technologies. The EC will hold a planning session after each workshop to address higher-level technical issues, compare progress against VGrADS research and EOT milestones, initiate and/or terminate projects, consider new target applications, revise research and EOT milestones as required to reflect new directions, and deal with issues of policy or resource allocation. The internal workshops and planning sessions embody the management strategy over short time periods: review, plan, and implement.

Initial VGrADS research and EOT milestones are listed in Table 1; this chart (and elaborations and revisions of it) will form the basis of our day-to-day and longer-term management. Similar charts are already used to organize the GrADS weekly teleconference, and this practice will continue. They will also be used by the EC in its discussions of high-level strategy.

**Software Management.** The proposed VGrADS budget includes funds for a senior software manager and a programmer. In addition to providing programming support for the VGrADS effort, these two software professionals will coordinate the overall VGrADS software effort among sites, produce VGrADS software distributions, and oversee the online repository of VGrADS source available from the VGrADS Web site. Both software professionals will attend the VGrADS technical teleconferences and internal workshops.

The software management team will be primarily responsible for maintaining the two public releases of VGrADS software planned during the project. VGrADS 1.0 is expected to include a prototype vgrid scheduler, the first version of VGrADSBank, a distributed binder/launcher, and a prototype mapper constructor based on clustering. VGrADS 2.0 is expected to include second-generation versions of all those, plus an integrated performance signature/temporal logic/statistical characterization toolkit for performance monitoring, novel fault-tolerance support, and a library optimizer and installer.

**Dissemination of Results.** VGrADS will disseminate research results through the standard academic venues (publication in journals and conferences, a VGrADS web site, external talks). VGrADS software artifacts will be available from a repository on the VGrADS Web site. VGrADS will host workshops and tutorials for the scientific community on topics related to VGrADS research. The workshops will bring VGrADS participants together with potential users, collaborators, and other researchers to discuss problems and progress in the area.

Table 1: Milestones for the VGrADS project

| | | Year 1 | Year 2 | Year 3 | Year 4 | Year 5 |
|---|---|---|---|---|---|---|
| **VGrADS Software Releases** | | | VGrADS 1.alpha (internal) | VGrADS 1.0 (public) | VGrADS 2.alpha (internal) | VGrADS 2.0 (public) |
| **Execution System** | **Virtualization** | • Prototype resource virtualization<br>• Virtual scheduling requirements study | • Prototype virtual scheduler (vsched)<br>• Novel vsched strategies for target apps | • Integrate vsched strategies into VGrADS<br>• Validate vsched on apps | • Evolve virtualization approaches based on applications experience | • Complete vsched validation on apps<br>• Extend to new apps |
| | **Performance** | • Temporal logic reasoning for contracts and signatures | • Genetic programming for performance control | • Initial tunable performance/fault-tolerance capabilities | • Integrated signature/ temporal logic/statistical characterization | • Extended application validation and assessment |
| | **Grid Economy** | • Dynamic commodity pricing for resources | • VGrADSBank (commodity pricing) for applications | • VGrADSBank stability and scalability studies | • Revised VGrADSBank based on studies | • Validate and extend VGrADSBank |
| | **Fault Tolerance** | • Experimental measurements of Grid & cluster reliability | • Prototype fault-tolerance library<br>• Non-transparent coordinated checkpoints | • Fault tolerance tested in applications<br>• Consider novel techniques | • Novel techniques (diskless checkpointing, caching, dynamic loading) developed | • Extend validation and assessment |
| **Programming Tools** | **Abstract Parallel Machine** | • Design virtualizing mapper constructor | • Prototype hybrid performance modeler<br>• Prototype virtual task graph clustering mapper | • Initial validation with applications | • Integration of VGrADSBank and fault-tolerance into mapper constructor | • Final validation with applications<br>• Extend to new apps |
| | **Abstract Component Machine** | • Design distributed binder/launcher<br>• Design workflow application support | • Build executables remotely with pre-installed libraries<br>• Cluster workflow map | • Prototype library installer<br>• Initial validation with applications | • Library installer with component tuning<br>• Execute COPs on heterogeneous nodes | • Library installer/optimizer validation on apps |
| **Applications** | | • Model, implement, and instrument apps on GrADSoft. | • Implement apps on VGrADS , compare to GrADSoft<br>• EOL study on vsched | • Stress test VGrADS 1.0 implementations<br>• MEAD study on abs. parallel machine | • Select second-generation applications<br>• GEMS study on perf toolkit, component mach | • Implement and evaluate apps on VGrADS 2.0<br>• Extend to new apps |
| **Broader Impacts** | **Education, Outreach, & Training** | • Host first VGrADS AGEP participants<br>• VGrADS participation in Hopper Conference | • VGrADS material added to AGEP, CS-CAMP, PACI programs<br>• VGrADS participation in Tapia Conference | • Rollout of AGEP to other sites<br>• VGrADS participation in Hopper Conference | • Expand VGrADS material in EOT progs.<br>• VGrADS participation in Tapia Conference | • Continue VGrADS AGEP experience<br>• VGrADS participation in Hopper Conference |
| | **Technology Transfer** | • SC tutorial on Grid computing,<br>• SC demonstrations of GrADSoft | • SC tutorial on Grid computing | • Migrate VGrADS ideas to mainstream Grid (e.g. OGSA, NMI)<br>• SC tutorial on VGrADS 1.0 | • SC tutorials. Topics TBD by public interest<br>• Approach industrial partners for VGrADS joint developments | • SC tutorial on VGrADS 2.0<br>• Migrate VGrADS ideas to mainstream Grid (e.g. OGSA, NMI) |