I. Project Activities

The “Computational Grid,” as described in *The Grid: Blueprint for a New Computing Infrastructure* and demonstrated by many proof-of-concept applications, promises to connect computers, databases, and people in a network to solve problems in scientific research and other diverse fields. However, the complexity, unreliability, and overhead of low-level operations in today's systems obscure the Grid's potential. The five-year Virtual Grid Application Development Software (VGrADS) project is attacking a fundamental part of this problem— how to more effectively program these highly complex and dynamic systems. It is developing software tools that simplify and accelerate the development of Grid applications and services, while delivering high levels of performance and resource efficiency. This improved usability will greatly expand the community of Grid users and developers. In the process, VGrADS will contribute to both the theory and practice of distributed computation.

To address these aims, VGrADS is exploring, defining, and implementing a hierarchy of virtual resources and a set of programming models for Grid computing. It is conducting research in three key areas:

1. Virtual Grid (VG) architectures, enabling a separation of concerns between high-level services and the Grid's inherent complexity. This architecture is implemented by the Virtual Grid Execution System (vgES).
2. Programming models, compilers, component libraries, and tools supporting creation of Grid applications.
3. Core software technologies, including performance-efficient scheduling, fault tolerance, and economic models for resource management, allowing scalable Grid computations.

VGrADS is pursuing this agenda by collaborating with leading scientific applications to elicit key challenges, validate results, and disseminate technology. It will distribute software that it creates in open-source form for the research community. It will also build on its PIs' past successes in human resource development by using existing programs to attract and retain women and minorities in computational science.

During the current reporting period (6/1/04 – 5/31/05), VGrADS research focused on the three inter-institutional efforts described in the following sections: *Applications, VGrADS Programming Tools,* and *VGrADS Execution System.* Project publications and additional information can be found at http://vgrads.rice.edu. The project Web site includes a participants-only area where VGrADS researchers exchange drafts of project documents and other materials.

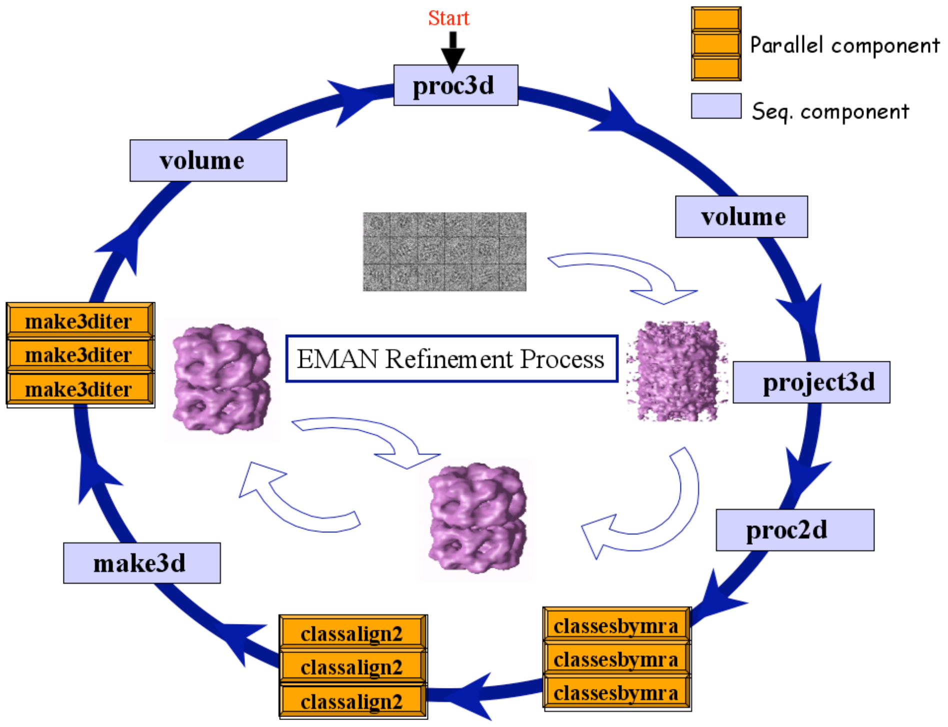
VGrADS includes researchers from Rice University; University of California, San Diego (UCSD); University of California, Santa Barbara (UCSB); University of Houston (UH); University of North Carolina (UNC); University of Southern California / Information Sciences Institute (USC); and University of Tennessee, Knoxville (UTK). Project design and coordination during the current reporting period were enabled through weekly technical teleconferences involving researchers from each of the VGrADS sites, six PI teleconferences, one PI meeting, a VGrADS planning workshop at UTK (9/13□14/04), and communication via VGrADS mailing lists. The 2005 VGrADS planning workshop will be held at UCSD (9/12□13/05). Research subproject participants met on a regular basis to exchange ideas and develop research plans; the EOT group at Rice met several times to develop and coordinate education and outreach activities. In addition, the National Science Foundation (NSF) reviewed the VGrADS project on April 28–29, 2005 at Rice University.

1 Applications (Rice, UCSD, UCSB, UH, UNC)

VGrADS research has always been driven by the needs of actual applications. Initially, we selected four applications (EMAN, EOL, GridSAT, and LEAD) based on our experience in the GrADS project and from other sources to help derive requirements for Virtual Grid (VG) functionality and to serve as tests of new tools methods. To date, they have been reasonably successful at helping to derive requirements for the VG functionality. We expect that as the Virtual Grid Execution System (vgES, see Section 3) is refined, moving the application implementations to the VG will both drive computer science research and produce domain science results from the applications themselves. We summarize recent work on each application in the following four subsections,

1.1 EMAN

The Rice and UH VGrADS application research efforts have focused on the EMAN software (http://ncmi.bcm.tmc.edu/ncmi/homes/stevel/EMAN/doc), which we take as a model for workflow-style applications. EMAN is a package for Electron Micrograph Analysis developed within the National Center for Macromolecular Imaging at Baylor College of Medicine (BCM) by Steve Ludtke, a senior researcher in Dr Wah Chiu's group. The package iteratively processes thousands to possibly tens of thousands of micrographs from electron microscopes in the determination of a macromolecular structure. The application and the iterative nature of the processing are illustrated in the following diagram.



As noted in previous reports, a key to porting EMAN (or any application) to the Grid is effective scheduling of tasks (particularly in the parallel phases) to available computational resources. Using the approach we pioneered in GrADS, we have created hybrid performance models for the application components and used these models in scheduling the components. The hybrid nature of the models comes from the need to manage multiple aspects of performance – UH researchers have derived an equation for the required computational operations based on knowledge of the component algorithms, while Rice researchers have used black-box instrumentation to derive models of memory access costs from benchmark runs. Finally, information from NWS is used to create a simple cost model for inter-task data movement. These models are added together to produce a single objective function, which our Grid scheduler attempts to minimize heuristically. Heuristics are required since the underlying problem (a variant of graph scheduling) is NP-complete.

Another recent addition to the EMAN application is the availability of Python scripting to control the refinement process. In fact, a non-trivial amount of our effort in the past year has been used to adapt our methods to this new capability and the resulting modest changes to the application workflow. To date, this has been mostly “engineering” rather than “fundamental research” work, but it positions VGrADS to move forward aggressively in the future. It also motivated the new Python compilation project mentioned in Section 2.

Using the scheduler described previously, we were able to demonstrate the largest Grid-enabled EMAN computation performed to date at SC2004. This computation used a testbed of 140 nodes in three clusters:

- 64 dual processor Itanium IA-64 nodes (900 MHz) at Rice University [RTC]
- 60 dual processor Itanium IA-64 nodes (1300 MHz) at UH [acrl]
- 16 Opteron nodes (2009 MHz) at UH [medusa]

The data set used was approximately 2GB of electron micrographs of the rdv virus particle. The 3D structure of the virus had been previously computed by EMAN on a different cluster at Baylor College of Medicine, allowing us to check our results for correctness. In summary, we found that our computation performance models were accurate in a relative sense:

- The model predicted that the “medusa” nodes were 3.41 times faster than the “RTC” nodes; the measured execution times showed that the “medusa” nodes were 3.82 times faster than the “RTC” nodes.
- The model predicted that the “medusa” nodes were 2.36 times faster than the “acrl” nodes; the measured execution times showed that the “medusa” nodes were 3.01 times faster than the “acrl” nodes.

This level of accuracy was sufficient to balance the load between the three clusters on the “classesbymra” component (which in practice dominates the time for the entire EMAN computation). For that phase of the computation, the scheduler mapped

- 1 task on some of the RTC nodes; others were idle (total time 383 min.)
- 1 task per acrl node (total time 308 min.)
- 3 tasks per medusa node (total time 410 min.)

Because all of the clusters were able to run simultaneously, the overall time was the maximum cluster time (410 min.). Note that this schedule is optimal for this problem; the other clusters were idle for much less than the time for a single task.

This experiment also gave us the opportunity to test the sensitivity of the scheduler to inaccuracies in the performance models. We report on those results in Section 2.

In the next year we plan to tackle a "challenge" problem with our Grid-enabled EMAN. Our collaborators at BCM have even larger data sets (involving the channel for the Ca+ ion), and the State of Texas has recently funded a large Grid activity (the TIGRE project, http://www.hipcat.net/Projects/tigre, and the LEARN network, http://www.tx-learn.org/). Preliminary results of the Ca+ channel data are already in press, but even small amounts of additional resolution (derived from additional iterative processes on extended data sets) would enable important new biological findings. We believe that the combination of VGrADS scheduling and the resources available through TIGRE will allow us to achieve additional resolution.

1.2 EOL

One of the applications initially identified by VGrADS researchers at UCSD as a research and development driver for the VGrADS project was the Encyclopedia of Life (EOL), a collaborative effort led by the San Diego Supercomputer Center for cataloging the complete genome of every living species. EOL is not a monolithic code, but rather a script (iGAP) that "glues" together a number of popular bioinformatics software packages and uses freely available biological sequence databases. iGAP provides the data and control flow among these packages and implements that is known as the "EOL pipeline". At the onset of the VGrADS project, EOL was deployed in production on a multi-institution grid, with two software technologies (APST and BSMW) for managing the deployment. Our broad goal was to demonstrate the benefit of our Virtual Grid (VG) abstraction and of the VGrADS approach in general, when applied to the EOL application.

As part of the evaluation and refinement of our ideas for VGs, the UCSD team has developed a simple resource abstraction for the EOL application. More specifically, we have developed a high-level model for the VG abstraction required to deploy EOL, as well as several refinements appropriate as the EOL implementation is enhanced for greater capability and efficiency. In addition to the macro-scale modeling, we have also engaged in detailed application behavioral modeling for subsets of the EOL computation – PSI-BLAST and 123D – which represent the major compute-intensive parts of the EOL pipeline. These models provide reasonable estimates for the computational work in each subtask and are being used to evaluate the value of precise dynamic resource information in efficient execution of these phases of EOL. In the end, our work with EOL has led to much the same findings as the ones we obtained with the EMAN application - the need for the homogeneous "cluster" and heterogeneous "bag of" resource abstractions and for coarse notions of network proximity when building VGs.

While this initial work with the EOL application was invaluable in defining and validating the VG concept, the initial plan of using EOL as a driver for VGrADS research and development activities beyond conceptual design has been reevaluated.

We have determined that working with the actual EOL software within the VGrADS project is no longer a viable option, for several reasons. First, the EOL software, i.e., the iGAP script, turned out to be an ad-hoc, intricate, and hardly maintainable code base, and thus most likely a counter-productive driver for our work. Second, the EOL team – being aware of the limitations of their current software – has been planning a complete refactoring of their code base. Third, while a refactoring of the software sounded promising, it turned out that funding for the EOL project has become problematic and its future is uncertain. Given both the inherent technical difficulties and the current uncertainty about this application, we have decided not to pursue work that entails development activity related to the EOL software per se. However, it is important to note that VGrADS technology is eminently applicable to applications in the same domain as EOL, i.e., bioinformatics applications that use sequence databases and sequence comparison tools for identifying matching sequences. Our current plan is to identify applications representative of the EOL domain for experimenting with the VGrADS technology. Our team has experience with applications in this domain, and in fact we used such an application in our work during the GrADS project. Furthermore, our preliminary work on EOL has provided us with additional experience with a number of bioinformatics applications that are also good candidates for VGrADS technology. So, in spite of this initial setback, we are confident that our work will be applicable and relevant to applications in the same class as EOL.

1.3 GridSAT

The UCSB VGrADS application work has focused on GridSAT, a parallel and complete Boolean satisfiability (SAT) solver used to solve non-trivial SAT problems in a grid environment. The application uses a parallel solver algorithm based on Chaff to (attempt to) solve SAT problems of the form ‘given a large, non-trivial Boolean expression, is there a variable configuration (and what are the variable values) which results in the expression evaluating to TRUE?’ The system stands apart from other SAT solvers in the sense that it was designed explicitly to run in grid environments, and has built in intelligent parallelism scheduling mechanisms. As a result of this design, the system has been used successfully and quickly to solve several previously unknown problems by utilizing vast amounts of computational resources.

GridSAT represents a grid application with resource requirements that are substantially different from EMAN and EOL. Since it was designed as a grid program from first principles (rather than as an adaptation of a parallel implementation) it includes many fault tolerance, latency tolerance, and resource-aware scheduling features that are necessary for grid application performance as explicit structural components. Thus, while it is code of some complexity, it represents the "high end" of grid application development as evidenced by its performance.

As a VGrADS driving application, GridSAT motivates both the functionality and the performance of the virtualized resource discovery and allocation mechanisms. The GridSAT scheduler considers resources abstractly, strictly in terms of their performance characteristics. VGs will be used to replace the GridSAT-internal abstraction mechanisms as a test of both their functionality (GridSAT should continue to function correctly) and their performance (the native GridSAT implementation is a “base” case). Moreover, GridSAT does not take advantage of virtualization in its native form. The virtualization features of VGrADS will enable GridSAT to scale beyond its current capabilities (currently thousands of processors), which again, forms the basis of an important empirical test of the VGrADS prototypes.

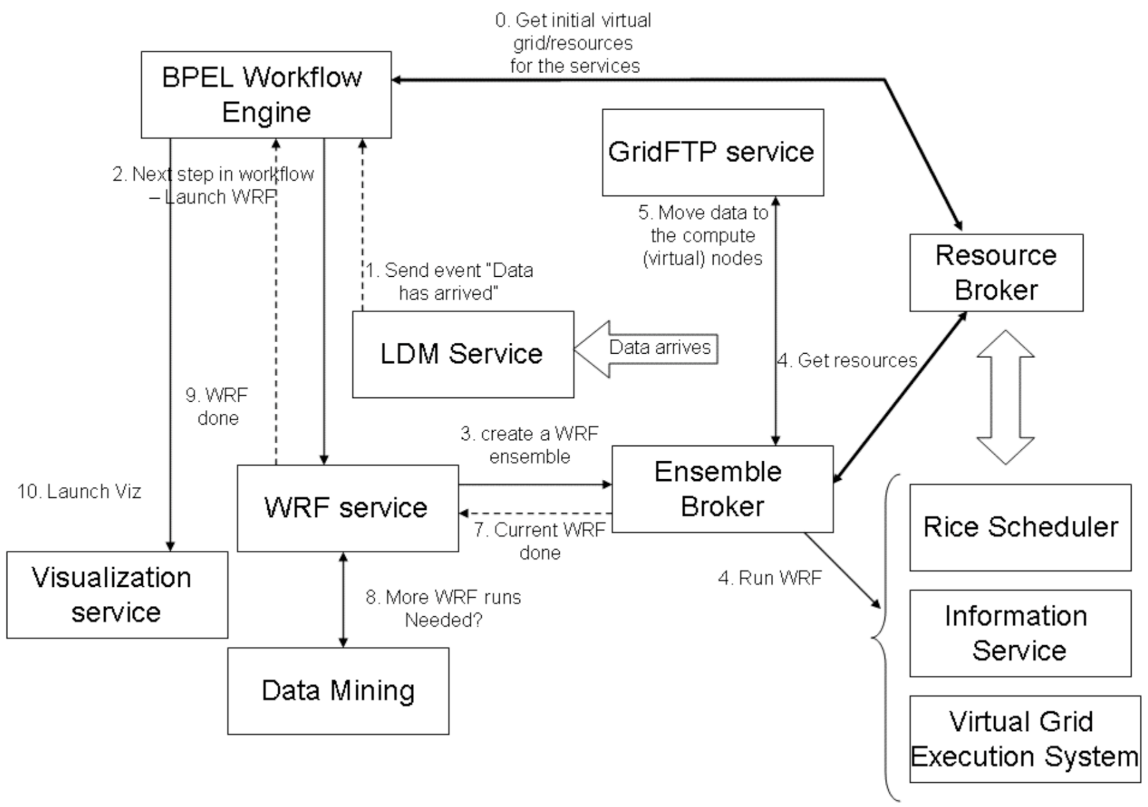
Finally, GridSAT's resource usage model is substantially more dynamic than that of the other VGrADS driving applications. It acquires resources only when it determines they will benefit execution (as opposed to having a maximal set specified when it is launched) and releases them as quickly as possible to prevent waste and promote allocation stability. To do so, the valuation of resources at any given moment in a GridSAT execution is related to the resources GridSAT is currently holding. This incremental form of resource discovery in which the currently held resources parameterize the resource search is unique among the VGrADS test codes, and motivates many of the dynamic features in the Virtual Grid Execution System (vgES) design outlined in Section 3.

1.4 LEAD

In response to the need for a national cyberinfrastructure in mesoscale meteorology, the UNC VGrADS research team (in collaboration with researchers at Oklahoma, UIUC, Alabama, Indiana, and elsewhere) is creating an integrated, scalable framework – known as Linked Environments for Atmospheric Discovery (LEAD) – for identifying, accessing, preparing, assimilating, predicting, managing, analyzing, mining, and visualizing a broad array of meteorological data and model output. The meteorology of LEAD is being developed under a separate ITR award (NSF 0315594); the VGrADS project collaborates with the LEAD project to apply VGrADS ideas and technology to the LEAD software.

A transforming element of LEAD is dynamic workflow orchestration and data management, which allows the use of analysis tools, forecast models, and data repositories not in fixed configurations or as static recipients of data, as is now the case, but rather as dynamically adaptive, on-demand, Grid-enabled systems that can (a) change configuration rapidly and automatically in response to weather; (b) continually be steered by new data; (c) respond to decision-driven inputs from users; (d) initiate other processes automatically; and (e) steer remote observing technologies to optimize data collection for the problem at hand. Toward these goals, LEAD research is focused on creating a series of interconnected, heterogeneous virtual IT “Grid environments” that are linked at several levels to enable data transport, service chaining, interoperability, and distributed computation.

We are working to understand LEAD workflow needs, and we are developing Virtual Grid (VG) specifications for the workflows, along with scheduling heuristics. One relatively simple sample workflow is shown in the following diagram.



(We should note that the definition of "workflow" in LEAD might be subtly different from the definition in EMAN or EOL. In particular, the services of LEAD are persistent whereas EMAN's tasks are not.)

We are also integrating measurement tools and infrastructure for system health and reliability, based on UNC's extant Health Application Programming Interface (HAPI) toolkit, with the UCSB Network Weather Service (NWS). More details of this effort can be found in Section 3 below.

Our preliminary analysis shows that LEAD has requirements much like EMAN and EOL for running an ensemble of simulations - the need for tightly connected clusters and loosely coordinated larger collections. In addition, there is a need for managing streaming data, as opposed to the fixed data collections used in the other two applications. LEAD is a service-oriented architecture. Every component in the LEAD architecture is encapsulated into individual services that represent the atomic application tasks as well as the resource and instrument monitoring agents that drive the workflow. Thus, in addition to the requirements similar to EMAN and EOL, for the LEAD application the VG needs to support the ability to select specific resources that run the persistent and transient services that need to be orchestrated by the workflow engine.

During the reporting period, our focus was on design of the interaction of the LEAD architecture with the VG API. The dynamic, adaptive and service-oriented workflows of LEAD require a VG with the identified persistent services at the beginning of the workflow execution. The VG expands and contracts as resource requirements change, accommodating changes in resource requirements driven by ensemble runs or weather phenomena. LEAD also allows interesting trade-offs between fidelity of results and fault tolerance, which require more investigation at both the application and system level before they can be refined into VG requirements. During the reporting period, UNC and Rice have been working collaboratively to construct and apply a performance model and simple heuristic-based scheduler (from Rice) to schedule simple LEAD workflows.

2 VGrADS Programming Tools (Rice, UH, USC)

The broad vision of the Programming Tools thrust is to provide for application users very high-level interfaces that allow automatic construction of capabilities that are (currently) hard to achieve in a Grid environment. Preliminary steps in this regard, reported last year, involved interpreting the results of previous GrADS experience for the purpose of helping to design the Virtual Grid (VG) abstractions. As interfaces to the Virtual Grid Execution System (vgES, see Section 3) have become more defined, we have begun design of tools to take advantage of this abstraction and tools to provide more application-specific abstractions.

More specifically, we have followed five research thrusts this year:

1. Improved scheduling for workflow computations on the Grid and VGs,
2. Performance prediction of application components to be mapped onto the Grid,
3. Compiling and optimizing node programs for use in a Grid environment,
4. Optimizing, launching, and controlling workflow applications on a Grid, and
5. Construction of Grid workflows from high-level scripts.

The subsections below discuss each in turn. Because the first release of vgES was not available until shortly before this report was written, none of these thrusts has yet been fully implemented.

2.1 Scheduling Workflow Applications

As previously reported, Rice and UH researchers have developed strategies for scheduling and executing workflow applications like EMAN (see Section 1) on Grid resources using the GradSoft infrastructure. These were described in some detail in our overview paper (http://vgrads.rice.edu/publications/ijpp-overview), which was supported by both the GrADS and VGrADS awards.

A workflow scheduler is fundamentally an optimization routine that attempts to minimize the execution time ("makespan") of a set of tasks on a given set of processors. For a workflow computation, the set of tasks are linked by producer-consumer communications; we therefore represent the application to be scheduled as a directed acyclic graph (DAG), with weights on the nodes representing computation and weights on the edges representing communication volume. Moreover, in a realistic Grid environment, different types of processors will have different computation times for the same task. (By "computation," we mean all operations in a single task, including memory access and file I/O, which are sometimes considered separately.) Similarly, communications time for a given amount of data will depend on which pair of processors are sending and receiving. In short, the weights in the DAG depend on the schedule, creating significant complexity. We therefore rely on heuristic scheduling strategies. Currently, we use three methods developed in other contexts: min-min, min-max, and sufferage. We compute these heuristic solutions and take the best available.

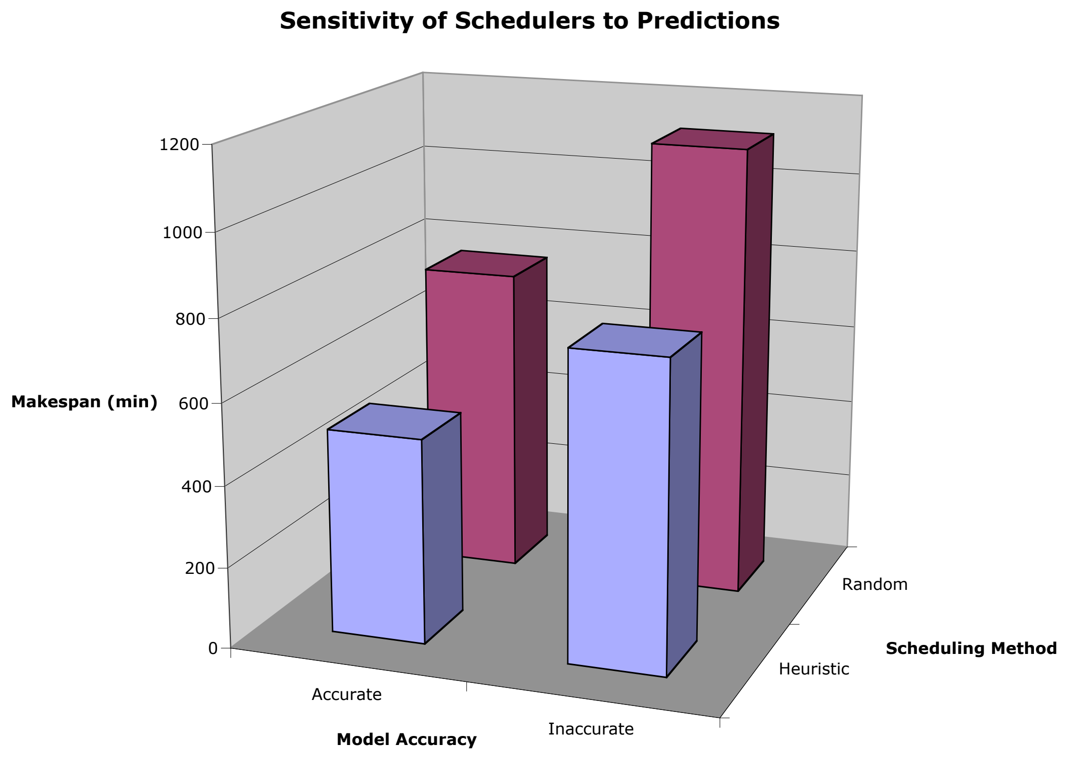
It bears mentioning that even the communication and computation times for known resources are often not known with precision, or are highly variable. We therefore use static proxies for the true times. We estimate the communication time from latency and bandwidth predictions delivered by the Network Weather Service (NWS). We estimate computation time from the performance prediction models described in Section 2.2. In future work, we intend to adjust the computation estimates (which we have validated for unloaded processors) by using NWS predictions of runtime load. Our scheduling method is independent of the details of these models, however.

We performed a number of experiments with this scheduler in the past year. We compared our heuristic scheduler with the standard approach in the Pegasus planner (see http://pegasus.isi.edu/) using the "Montage" (see http://montage.ipac.caltech.edu/) workflows. Execution times improved by more than 20% for homogeneous platforms, and by an order of magnitude (more than 20 times) for heterogeneous platforms. We used the scheduler on the largest Grid-enabled EMAN computation performed to date. Results of this experiment are described in Section 1.

In one of our preliminary runs for the EMAN experiment, a portion of the nodes experienced heavy load due to competing jobs. This motivated us to examine the sensitivity of our scheduler to inaccuracy in the performance model. We produced four runs of the same data set (the large "rdv" data mentioned in Section 1) on two of the clusters (the RTC at Rice and medusa at UH) varying two components of the scheduler:

- Using the VGrADS scheduler ("Heuristic" on the following chart) or the Pegasus randomized scheduler ("Random" on the following chart).
- Using the VGrADS performance model ("Accurate" on the following chart) or a known inaccurate performance model ("Inaccurate" on the following chart).

Since the randomized scheduler does not use performance models *per se*, we weighted the probability of choosing a given processor by that processor's estimated performance; a processor that ran three times as fast was three times as likely to be assigned a given task. The "inaccurate" model was based solely on CPU frequency, which gives a relative error of about 50% on EMAN for these clusters. (It makes the RTC appear 50% better than it is.) The results are graphed in the following chart. In summary, the performance model inaccuracy led to a performance decrease of about 50% under both schedulers due to induced load imbalance.



We are currently expanding the scheduler work in several ways:

- Researchers at Rice and UCSD are exploring new schedulers that explicitly use the hierarchical structure of the Virtual Grid (VG). In particular, we believe that using "rough" performance estimates to select resources will allow scalability, while more refined estimates (like the current VGrADS models) will still be suitable for scheduling on a smaller grid. This idea is developed in more detail in Section 3.
- In cooperation with the Texas Internet Grid for Research and Education (TIGRE, http://www.hipcat.net/Projects/tigre/), Rice and UH researchers are beginning work on a "challenge" EMAN problem that will both produce new discipline science and test the scalability of our scheduler. We estimate that approximately 3000 processors will be needed, forcing our scheduler to handle much more data than previously. Also, the workflow for the problem will be more complex due to algorithmic improvements, further testing the scheduler's capabilities.

2.2 Automatic Construction of Performance Models

One aspect of making applications Grid aware, in the sense of our current infrastructure, is the construction of performance models for the application's components in order to make proper decisions for resource selection and scheduling. The construction of performance models for the EMAN application has been a major focus of research efforts at both Rice and UH.

Beyond EMAN, researchers at Rice have pursued an approach that makes use of a set of equations, the structure of which is derived from knowledge of the component's computational requirements and the dependence upon input variables such as micrograph sizes, number of micrographs, and variables controlling the processing of the micrographs. Model parameters are then determined from execution traces. The objective is to attempt to separate application characteristics invariant with respect to architectures from those that are dependent. For instance, the number of floating point operations should be independent of the platform used for the processing, but the number of instructions will depend upon both that platform and the compiler used, as well as compiler options used. Similarly, memory access patterns should have a strong correlation to the application, while the number of cycles required will depend significantly on the memory system.

To better understand the impact of the memory system on performance, we have been pursuing a novel approach that uses application binary analysis to instrument preliminary runs of the application in order to determine the distance, in memory accesses, between accesses to the same cache block. Recent results of this approach were published in SIGMETRICS in 2004 showing that, for a set of six scientific applications, this methodology usually produces estimates within 10% of the measured running time. Of the two applications that predict poorly, both may be due to inaccuracy in estimating the cost of L2 cache misses, although more investigation is warranted.

By integrating the components described, we have derived, validated, and integrated into the scheduler a model for the main EMAN components. We are now in the process of automating the model fitting and investigating the sensitivity of the results to compilers being used, as well as the impact of compiler options being used. The main platforms targeted at the moment are Opteron and Itanium2 based clusters.

2.3 Compiling and Optimizing Node Programs

Researchers at Rice have pursued methods for compilation on individual Grid nodes under the aegis of VGrADS programming tools. This effort aims to support efficiency in the basic components of Grid applications in two ways.

Heterogeneity in the Virtual Grid (VG) — One goal of the VGrADS project is to create a set of tools that let the programmer target an abstract, or virtual, grid and let the programmer ignore the specific physical machines on which the code will execute. This virtualization should include the ability to run the code on a heterogeneous collection of machines, specifically, machines with different processor architectures. (In the GrADS project, the level of heterogeneity supported was varying Intel Pentium models with different amounts of RAM.)

To support processor heterogeneity, we need a compiler that can both target multiple architectures and serve as an experimental platform. While most of the experimentation in VGrADS (and the earlier GrADS) project has used GCC as its compiler, GCC is a particularly poor choice for experimentation in both compile-time optimization and runtime re-optimization. After evaluating several alternatives, we believe that the LLVM system (from Adve's group at Illinois) is a good platform for our experiments. Accordingly, we have worked on improving LLVM's base performance by improving the quality of its code generation. We implemented a pair of new register allocators for LLVM and are building an aggressive instruction scheduler for it. Because of LLVM's careful attention to retargetability, the allocators work on multiple architectures with almost no change. As a result of this work, we expect LLVM to be a viable compiler for use in VGrADS experiments, which will, in turn, let us experiment directly with runtime re-optimization. We will distribute the new LLVM passes that we have produced as soon as we clear several potential legal hurdles.

Runtime Re-optimization — Single-processor underperformance is a significant challenge for any Grid-based computation. In practice, underperformance of one or more processors can nullify the results of scheduling and, in extreme cases, of running the task in parallel. One way that the runtime system can respond to underperformance is to re-optimize the program in ways that capitalize on knowledge that was not available at compile time.

Runtime re-optimization (and its intellectual cousin, just-in-time compilation) has received a great deal of attention in the literature over the last ten years. The ideas have proven themselves profitable in high-overhead environments, such as a Java virtual machine. In low-overhead scientific environments, it is harder to demonstrate actual improvements from runtime code rewriting. Working inside LLVM, we are developing a strategy for statically-planned, dynamically-applied optimization. These algorithms involve compile-time planning for alternative code sequences and runtime rewriting to implement these alternative code sequences in response to poor performance. This approach has the potential to achieve some of the benefits and flexibility of runtime re-optimization while imposing a much lower runtime cost. Anshuman DasGupta presented a Ph.D. thesis proposal on this topic in May 2005. The initial set of algorithms will be implemented during the 2005–2006 academic year.

2.4 Managing Workflow Applications

Successfully executing an application on the Grid requires a framework for managing that execution. In the GrADS project, we developed the AppMgr (Application Manager), a software system that undertook the scheduling and mapping interactions with the GrADS execution system, launched a performance monitor for the application, staged program and data files as needed, launched the individual tasks of the application and linked them to the monitor, gathered and responded to signals from the monitor (e.g. for adaptivity), detected application termination, and performed final clean-up. As VGrADS has changed and simplified some of these tasks with vgES, it has been necessary to rethink this design.

In order to leverage outside research as much as possible in the application manager, we are considering adopting (in whole or in part) the Pegasus (http://pegasus.isi.edu/) system developed at USC as the basis for VGrADS work. Pegasus is a workflow management system built on the DAGMAN (http://www.cs.wisc.edu/condor/dagman/) meta-scheduler for the Condor (http://www.cs.wisc.edu/condor/) system. Pegasus takes as input “abstract” workflows (i.e. workflow DAGs with no execution resources assigned), optimizes the DAG to build a “concrete” workflow (i.e. a DAG, possibly restructured, with resources for all nodes assigned), and passes it to DAGMAN for execution on a Condor pool. Pegasus thereby inherits many useful properties of Condor—most notably support for fault tolerance—but is not so limited in certain respects—most notably in scheduling operations.

So far we have just begun explorations, but the signs are good. USC and Rice collaborated to integrate heuristic schedulers into Pegasus, leading to the comparison of the VGrADS scheduler and randomized methods reported in Section 2.1. Work at USC was already in progress to optimize workflows for Grid execution, and it appears that those techniques are complementary to the scheduling methods already under development at Rice and UCSD. The USC team successfully ported the EMAN application (described in Section 1) to Pegasus and ran it on the USC Condor pool as a proof of concept. At this writing, the USC and Rice teams are working to retarget the DAGMAN launch process to use vgES rather than lower-level services such as Globus GRAM. It is unclear whether this is best done by incorporating an interface from Condor to vgES, or by modifying DAGMAN to use vgES more directly.

2.5 Supporting High-Level Scripting

Because many Grid-enabled workflow applications (including the next release of EMAN) are driven by high-level scripts, Rice VGrADS researchers have begun research on tools to support Python and other scripting languages. In the context of the Grid, this may include automatic generation of workflow from a Python script describing the application. At a finer granularity, we also plan to work on automated distribution and alignment of data on parallel machines and clusters. However, solving all of these problems would, we believe, involve type inference techniques. Our research group has made significant progress in type inference techniques for scripting languages such as Matlab as part of the telescoping languages project, and we hope that the same technology, with some modifications, may be applied to Python for Grid problems.

3 VGrADS Execution System (UCSD, UCSB, UTK, UNC)

3.1 Development of Virtual Grid (VG) Abstractions and Prototypes

The core activity of the execution system team, led by the UCSD team, is to develop and refine the notion of Virtual Grids (VGs). The core elements include the VG abstractions, implementation technologies, and underlying statistical resource classification techniques. To effectively support applications while achieving efficient resource selection and scheduling, VGs must enable applications to express simply their desired resource abstraction and preferences. To develop these abstractions, we have pursued a systematic study of several leading grid application efforts – Encyclopedia of Life (UCSD), EMAN (UH, Rice), LEAD (UNC), and SATisfiability (UCSB) – developing for each of them a simple VG description that expresses desired resource abstraction. The UCSD group is spearheading a large team effort to evaluate these abstractions for appropriateness for application expression and optimization. Major achievements in understanding and realizing the idea of VGs in the current year include:

- Definition of the Virtual Grid Description Language (vgDL) Version 0.95 design, October 2004

- Design and implementation of the Virtual Grid Execution System (vgES): System Architecture and Component, November 2004
- Design and implementation of a vgES prototype, which implements static VGs, providing a computer science research quality software infrastructure for VGrADS, March 2005
- Definition and implementation of the VG Selection and Runtime Attribute Systems, which provide applications the ability to construct VGs, and conveniently access both static and dynamic resource information using the VG resource abstraction, March 2005

We describe the technical insights and advances inherent in each of these areas in turn below.

3.1.1 Definition of the Virtual Grid Description Language (vgDL)

A central element of the VG approach is a resource description language (vgDL) based on application-centric resource abstractions. The vgDL design is based on detailed study of four real-world grid applications (EMAN, EOL, LEAD, GridSAT) conducted in Year 1 of the project. In this study, we identified the high-level resource abstractions used by these grid applications, as well as a range of typical preferences and attributes for the resources ultimately used to implement these resource abstractions. **The key insight is that sophisticated application developers of grid applications typically use a high-level resource abstraction to organize their application management for performance, reliability, etc.** Our design of vgDL provides application developers a convenient high-level means to describe the high-level abstractions explicitly, and then allows the application to use a concrete realization of that resource abstraction in the form of a VG. The vgDL language is designed to be easily readable for programs, and it enables specification of both qualitative and quantitative resource requirements. The key strength of vgDL is that it not only uses simple resources abstractions, supporting simple specifications, but that it is also a rich, expressive language that enables experts to control resource specification with precision. For a full description of the language and its design, see http://vgrads.rice.edu/publications/vgdl-tr. Key ideas in the design of the vgDL language are discussed in the following named sections.

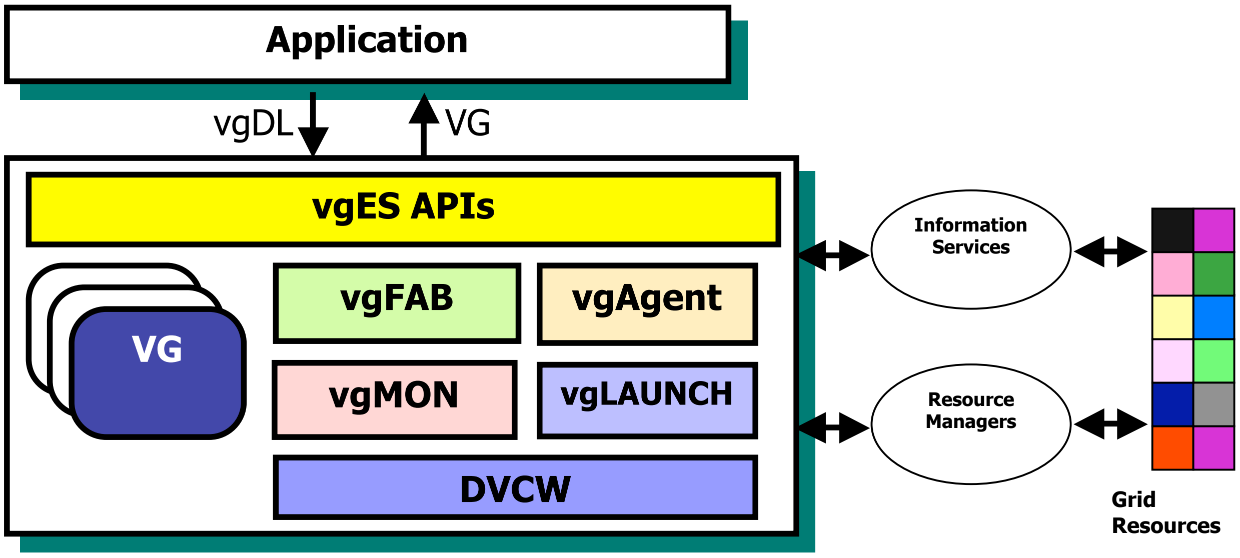
Resource Aggregators: Our study of grid applications shows that most applications actually use resource abstractions that are simple aggregations of resources that in turn have simple relationships in terms of their properties (i.e., all same CPU type, all >1GB memory, etc.). **The prevalent use of simple aggregator structures for resource abstractions reflects a desire for applications to simplify their interaction with the resources and resource environment as the focus is on effective realization of the already complex application.** To meet this need, vgDL includes only three aggregators: “LooseBags, TightBags” (loosely-connected and tightly-connected collections of heterogeneous resources respectively), and “Clusters” (tightly-connected homogeneous collections of nodes). Applications can compose these aggregators (e.g., a “LooseBag” of “Clusters”). Aggregator properties, and many of the characteristics of vgDL descriptions are *qualitative*. In our analysis of application needs, we found that for many applications, detailed quantitative specifications are a distraction, and as such caused resource specifications and the resulting application programs to be fragile. We believe our qualitative approach will enable many applications to construct simple, robust, and enduring vgDL specifications. Of course, aggregators build on traditional definitions of individual resources.

Network Connectivity: For many applications, inter-node and inter-aggregate network connectivity is critical for performance. vgDL includes four operators that define network connectivity: `"close", "far", "highBW", and "lowBW".` These composers indicate coarse, qualitative notions of network proximity in terms of latency and bandwidth that are intentionally vague. Each specific vgDL implementation will use specific default values for defining quantitative meanings. We believe that appropriate defaults are likely to be an ongoing research topic, and may not only change as technology evolves, but also be a function of the resource environment in which a vgDL request is viewed. The design rationale for this approach is again that, for many applications, detailed quantitative specifications are a distraction, and a detriment to simple, robust, and enduring vgDL specifications. Applications that require detailed quantitative measurements to optimize execution can of course obtain this information by querying the VG attribute system once a VG has been instantiated. Such information can be used to apply quantitatively optimized scheduling algorithms and other optimizations**.**

Approximate Selection and Scalar Ranking: By designing vgDL to approximately specify the needed resources, our intent is also to reflect the difficulty of achieving good, much less optimal selection and binding of resources in the open, shared, competitive resource environments that typify most production grids. In addition, future resource environments are likely to be much larger than the thousands of resources that comprise grids today. Future systems are likely to have millions of distinct resources. One future research focus is the design of resource selection and binding algorithms and mechanisms that achieve provably good results in the face of resource competition and in very large resource environments (millions of resources). Because applications will generally only provide approximate resource requests, vgDL also enables applications to express preference using a scalar *ranking function.* The ranking function is a user-defined expression consisting of basic arithmetic operations, which produces a scalar value that represents desirability. The ranking function is used by the vgES to rank the candidates that meet the vgDL specification constraints to determine which one the application would prefer. If no ranking function is provided, or the ranking function does not distinguish between the candidates, a set of default preferences is used.

3.1.2 Design and Implementation of the Virtual Grid Execution System (vgES): System Architecture and Component Design

Concurrently with the vgDL design, we worked in a collaborative fashion with the entire VGrADS team to develop the vgES architecture. This design is critical both to the notion of VGs – defining the two major interfaces between the VGrADS execution system and the programming tools / applications. In addition, it defines the componentization of the internals of vgES, forming a framework for innovation in the research infrastructure by researchers across the entire VGrADS execution system team. The critical aspects of the vgES system architecture are defined in the remainder of this section. The system diagram and major components are depicted as follows:



The two major vgES application interfaces include: 1) the VG creation and management API's through which the application submits a vgDL description, asks the vgES to create the VG, and receives the constructed VG, and 2) the VG itself, which provides an explicit, hierarchical representation of the resources that have been bound into the VG, and also presents all static and dynamic information about those resources through an extensible attribute-based interface. The VG creation and management interface includes operations for extending, curtailing, and even merging a VG dynamically. The VG interface provides primitives for navigating the VG structure, defining new attributes, reading attributes, and incorporating new information providers. Together these two interfaces provide a simple, expressive interface for grid resource management.

The major elements of the vgES system architecture are implemented in five major components.

- **vgFAB** is a "finder and binder" that performs integrated resource selection and binding. The vgFAB is the primary implementer of vgDL and encompasses the creation of the VG abstraction.
- **vgLaunch** is an application launcher that initiates the application on the resources bound by vgFAB. Future implementations may leverage scalable launchers from other research efforts.
- **DVCW** is the module that encapsulates the DVC (developed as part of the OptIPuter project), which in turn encapsulates the standard resource management interfaces provided by Globus.
- **vgAgent** provides a critical interface function, retrieving static/dynamic resource information from existing information services systems and supporting the finding and binding in vgFAB, as well as the application access to static and dynamic resource information in the VG. Currently, this system provides full access to MDS and NWS data.
- **vgMON** is a distributed monitoring component that tracks resource behavior against application expectations, notifying the application when things vary significantly from expected behavior.

These five components can each be implemented locally, or as a distributed service. In addition, this clean factorization enables research and innovation in each of these areas (and above the vgES interface) with moderate effort. This ensures that vgES serves the important role of providing a research platform for efforts to explore research issues within vgES and those properly framed outside of it.

3.1.3 Design and Implementation of a vgES prototype

We have completed a first implementation of vgES (Version 0.7), including basic testing and documentation, culminating in an internal VGrADS project release on March 21, 2005. The key capabilities of this early research prototype release are:

- vgFAB
  - VG creation and termination: createVG(), terminateVG()
  - Application launch and control: runCmd(), done(), terminate()
  - Information/Attributes: readAttribute(),getAttributes(), writeAttribute(), readInterAttribute()
  - VG navigation: getNumChildren(), getParent(), getChild()
  - File transfer: copyFromNode(), copyToNode()

- vgAgent
    - Generic interface for periodic updates to vgFAB from information providers, resource classification
    - Generic interface for periodic and “on demand” computation of VG attributes, allowing connection to wide range of dynamic information providers
    - Implementation of modules for both systems for MDS
    - Implementation of modules for NWS
- DVCW
    - Resource binding and release
    - Basic job control and monitoring
    - Support for Globus 3.2.1
    - Improved packaging and deployment for underlying DVC

We expect to implement the major additional defined functionality (dynamic VG management capabilities, initial vgMON implementation) before the end of calendar year 2005, but this goal is subject to revision based on emerging research objectives and resource constraints. Because this is an ongoing research system, each of the vgES components is the subject of active research and will likely be reimplemented in several ways to explore new algorithms, grid strategies, or to reflect application needs and perform systematic research experiments.

The vgES Version 0.7 system implements a Java API only and is implemented as a standalone Java program. The vgES Version 0.7 prototype is intended for use by essentially all parts of the VGrADS team to enable their research experiments. We have received numerous inquiries from large-scale application and infrastructure projects expressing interest in use of vgES in broader settings. However, given the fact that VGrADS is a computer science systems research project, with no charter (and insufficient resources) to produce software infrastructure for broader use, such activities will necessarily be limited unless more resources can be obtained.

3.1.4 Definition and Implementation of the VG Selection and Runtime Attribute Systems

The vgES system includes two types of resource attributes. The first type, selection attributes, are used by the vgFAB to implement the vgDL language, and guide the selection and binding of resources to form the VG in accord with the application request. These selection attributes include traditional resource attributes such as CPU type, speed, memory, disk available, etc. vgES also includes a second type of resource attribute, used at runtime in the VG resource abstraction. The purpose of these attributes is to provide convenient information access for the application through the VG abstraction. This situation is depicted in Figure 1.

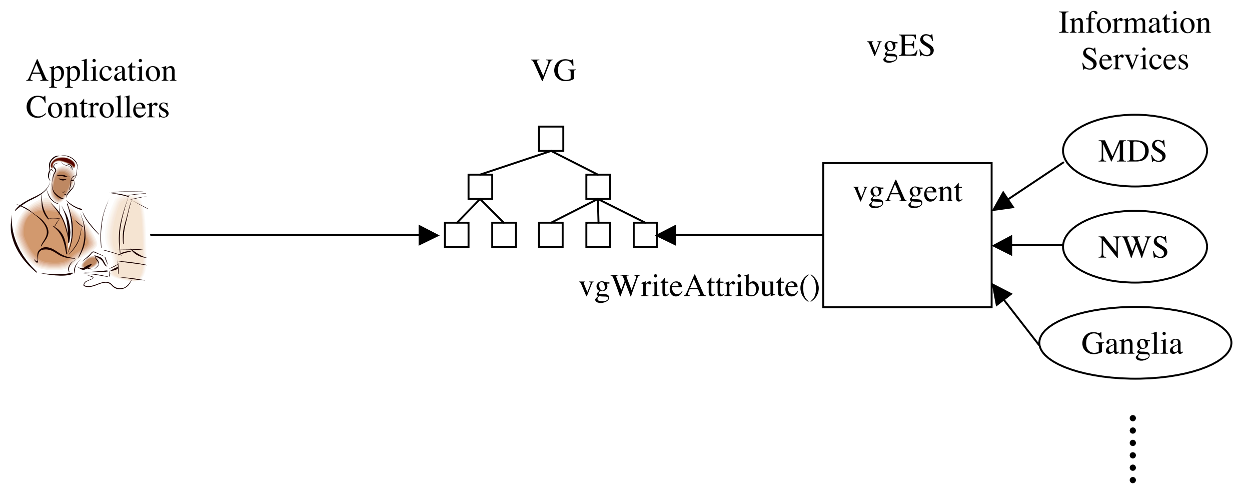


Figure 1: VG resource attributes connect application controllers (applications), VGs, and a collection of information services. Application controllers use vgReadAttribute() to access information. Additional attributes can be defined either by use of vgWriteAttribute() or vgAgent.

In the reporting period, we have completed designs and implementations of both of these resource attribute systems in the form of the vgAgent and vgFAB. These implementations define the standard attributes and how applications and information providers can flexibly extend the set of attributes. In addition, in the vgAgent, several update models (variants on traditional push and pull models) have been explored and are implemented. Together, the base system implementation and extensibility we believe comprises a basic and powerful research substrate for a wide range of experiments in dynamic resource management for applications as contemplated in the core VGrADS research agenda.

In the following sections, we discuss several other important research accomplishments and directions involving the use, development, and evaluation of VGs: 1) automatic resource characterization, 2) application studies, 3) techniques for fault-tolerance, 4) scheduling for VGs, and 5) more general reasoning about application properties.

3.2 Automatic Resource Characterization

Achieving automatic resource characterization is a critical capability to enabling good resource selection, efficient use, management for fault tolerance, etc. These efforts, led by the UCSB team, have produced several significant results this year, including:

- Development of new statistical techniques for predicting and quantitatively characterizing machine availability, October 2004
- Development of new statistical techniques for predicting batch-queue wait times, March 2005
- Development of prototype automatic checkpoint scheduler, December 2004
- Development of high-performance and scalable Virtual Grid (VG) interface to the NWS for delivery of resource characterizations, March 2005

The statistical results focus on a new set of forecasting methods that produce verifiable confidence bounds for both machine availability and batch-queue waiting times. Availability predictions are critical to the development of performance efficient fault-tolerance mechanisms. With a quantitatively accurate prediction of host availability, it is possible to determine both effective checkpointing strategies that adapt to current conditions, and effective replication strategies that can then be virtualized into a single resource by the vgES. Of course, availability data for resources can generally be included in the vgES resource selection and VG construction process.

To determine the effectiveness of our new techniques, we also developed an optimal checkpoint scheduler that automatically determines the durations between checkpoints that optimize application execution for a given resource. The system uses NWS machine availability sensors to constantly gather and record quantitative machine reliability measures. It then automatically fits a series of parametric models to the data, and evaluates the fitness of each model. The best model among the ones available is chosen and the checkpoint schedule is then generated. In addition to improving application performance in comparison to previous checkpointing schemes, our scheduler also dramatically reduces the network load generated by regular checkpoints that must be stored remotely, thereby reducing network contention.

In a similar vein, we have developed a technique for automatically producing a confidence range for the time a particular job will wait in a given batch queue. While we have not developed a deployable prototype (as we have for the checkpoint scheduling system), we plan to do so during the next reporting period. We have verified the technique, however, using archival logging data spanning ten years at various NSF and DOE sites—an empirical evaluation that consists of over 1,000,000 predictions.

Another innovation that we have developed is a scalable virtualization system that is capable of producing a quantitative "picture" of the grid resource pool. By automatically sensing the topology of networked resources, the system constructs a virtualization by eliding redundant data and replacing it with automatically generated forecasts. For example, to produce a map of the network connectivity between hosts at UCSB and those at UTK, the system determines that the campus-to-campus network dominates the performance any host at one campus observes when communicating with any host at the other campus. Thus the campus-to-campus connectivity need only be measured once and the forecasts used for any pair of hosts. Because the forecasting can take place either inside the NWS or in the client API library, this system can deliver data to the vgES rapidly and scalably. We have designed and implemented a vgES–NWS interface that allows the wealth of NWS information to be concisely transferred to both the vgFAB (guiding resource selection) and to individual VGs (enabling applications to manage their use across resources in their VG).

Note that the performance component for this virtualization is critical. While the vgDL resource specification layer hides many of the quantitative details to aid in the simplicity of programming, high-performance and scalable applications such as GridSAT often require effective resource characterization in quantitative terms. Thus, the VG snapshot must be assembled and distributed so as not to impact application performance while capturing grids at scale. At present, our system has been tested with 10,000 forecasts, which it assembles and delivers in approximately 6 seconds. While this prototype is encouraging and will meet the needs of the VGrADS project, we are, at present, enhancing both the scalability and performance of the system to enhance its future utility.

3.3 Fault Tolerant Computing via FT-MPI and GridSolve

We are implementing a strategy for the use of iterative methods under the Virtual Grid Execution System (vgES). These efforts, led by the UTK team, will use our FT-LA (fault-tolerant linear algebra) Package together with FT-MPI and GridSolve. FT-MPI is a fault tolerant implementation of the MPI 1.2 specification being developed at UTK. The approach to adding fault tolerance is to enhance the error modes of MPI to allow damaged communicators to be repaired. The UTK team is exploring the use of FT-MPI as a mechanism for adding fault tolerance to MPI applications that run under vgES. The initial implementation is via the GridSolve framework to VGrADS. GridSolve is middleware created to provide a seamless bridge between the simple, standard programming interfaces, such as Fortran, C, and Matlab, and desktop Scientific Computing Environments (SCEs).

FT-LA will analyze the matrix to determine the method and the approximate number of iterations it will take to converge. (It does this via an effort we have called SALSA.) Given the approximate number of iterations, the number of operations per iteration for the specific matrix and method, and the time constraint, FT-LA will call vgES to construct an appropriate Virtual Grid (VG). (in this case, a set of processors to solve the problem, most likely within a cluster). vgES will return a set of processors with information on the predicted number of process failures that will occur during the execution time. This predicted failure rate might, for example, be provided by the automatic resource characterization system. FT-LA will set up the problem and use the Virtual Grid Execution System (vgES) to launch the job on the processors. The application will run to completion in the presence of a number of process failures. The system will be monitored and, if required, the application can be stopped and restarted on another configuration.

3.4 VGrADS Scheduling

One of the challenges for achieving high levels of performance when deploying grid applications is that of *scheduling*, which is the decision process by which application components (computation, communication, data) are assigned to grid resources (CPUs, network, storage). Scheduling is a well-known difficult problem in computer science and has been studied by many researchers in the context of grid applications and grid platforms. The Virtual Grid (VG) abstraction developed in this project provides a novel and powerful way to approach application scheduling. Indeed, this abstraction decouples the scheduling activity from resource discovery, resource selection, and resource binding. Decoupling scheduling from resource discovery is a natural idea that has been validated in several research works, including our own during the GrADS project, but we anticipate that decoupling from resource selection and resource binding is key to enable effective application scheduling in emerging large-scale grids.

While scheduling heuristics are typically designed to exhibit polynomial complexity, the actual running time of a heuristic, i.e., the time to compute an application schedule, can account for a significant fraction of the application's execution time for large pools of resources (especially when network connectivity is taken into account by the heuristic). Therefore, the traditional grid application scheduling approach, which consists in running a heuristic over the entire resource universe and thereby obtaining an "implicit" resource selection, will often not scale to large systems. By contrast, the VGrADS approach lets the user/application specify its resource needs and, using a structured resource information database, performs fast and accurate resource selection to generate a VG. At that point, the scheduling heuristic can be executed just over the resources within the VG, which is exponentially less costly than looking at the whole resource universe, and in fact opens the door to investigating the use of more costly but more effective heuristics.

In addition to decoupling resource selection from scheduling, the VG abstraction also decouples resource binding from scheduling since resources in a VG are already bound to the application when the VG is instantiated. This makes it possible to have the vgES deal with the diversity of local resource managers and resource management policies, as well as with unreachable resources, so that application scheduling can be done with simple resource access and availability models. This is radically different from state-of-the-art application scheduling and is another major capability for enabling effective grid application scheduling in large-scale systems.

The UCSD and Rice teams are currently running simulation experiments on synthetic large-scale grid systems (generated with the grid resource generator developed in Year 1 of this project) in a view to validate and quantify the benefit of grid application scheduling with the VG abstraction in terms of decoupling application scheduling both from resource selection and from resource binding.

3.5 Workflows, Monitoring and Reasoning About Behavior

The UNC VGrADS research team (in collaboration with researchers at Oklahoma, UIUC, Alabama, Indiana, and elsewhere) is creating an integrated, scalable framework—Linked Environments for Atmospheric Discovery (LEAD)—for identifying, accessing, preparing, assimilating, predicting, managing, analyzing, mining, and visualizing a broad array of meteorological data and model output. The LEAD toolkit is being developed under a separate ITR award (NSF 03-15594), where research is focused on creating a series of interconnected, heterogeneous virtual IT "Grid environments" that are linked at several levels to enable data transport, service chaining, interoperability, and distributed computation.

As part of the VGrADS funded activities, we are engaged in three activities:
(a) collaborating with LEAD to apply VGrADS ideas and technology to the LEAD software stack, (b) integrating fault monitoring infrastructure and techniques with the VGrADS toolkit, and (c) developing general reasoning mechanisms to assess application performance with respect to contacts and desired behavior.

In the reporting period, we have advanced our understanding of the resource needs of LEAD workflow, and we have developed a Virtual Grid (VG) specification for the workflows. This aspect of our work is described in more detail in Section 1.

Because the VG execution system will use the Network Weather Service (NWS) to collect resource performance data, we are integrating measurement tools and infrastructure for system health and reliability, based on UNC's extant Health Application Programming Interface (HAPI) toolkit with NWS. The HAPI package is used for discovery and monitoring of health-related diagnostic information on Linux clusters. This is intended to support VGs that provide fault tolerance by tunable grid specifications (e.g., by redundantly and invisibly scheduling extra copies of tasks to increase the probability of task completion).

Finally, we are developing a qualitative temporal reasoning framework to support performance validation for VGs. Our goal is to reason about temporal events to differentiate between severe, persistent behavioral violations and transient ones. This will help bind the expectations of applications with the resource behavior in the VG execution system. Our framework has three major components: (a) classification of temporal behaviors observed in scientific workloads, (b) development of a temporal algebra for temporal behaviors and interactions and (c) a methodology for predicting behavior interactions.

We are collecting system-level, quantitative resource usage data and are analyzing the data using statistical and clustering techniques. The results define a set of high-level, qualitative behavioral phases based on resource type (e.g.: CPU-intensive, IO-intensive, and CPU+IO intensive) and resource usage pattern (e.g.: oscillatory or steady). We have developed a temporal algebra that expresses phase behaviors, along with a methodology for predicting the phases that result from the convolutions of phases.

II. Findings

During the reporting period (6/1/04–5/31/05), VGrADS research focused on three inter-institutional efforts: *Applications, VGrADS Programming Tools,* and *VGrADS Execution System*. A major focus for all three efforts in this reporting period was to refine the Virtual Grid (VG) concept and implement a first prototype of the Virtual Grid Execution System (vgES). The following sections summarize the findings of each subproject.

1 Applications (Rice, UCSD, UCSB, UH, UNC)

Work on applications during this reporting period focused on determining requirements for VGs to support common programming practice and experimenting with improved methods for executing grid-enabled applications.

EMAN: EMAN showed that support for "clusters" (collections of relatively homogeneous processors, with a relatively homogeneous connectivity and a common file system) was required for applications with tightly coupled components. Higher-level "bags" (collections of heterogeneous resources) are also appropriate for systems built from such components, such as parameter sweeps. Finally, it appears that multi-level scheduling schemes, like our current distinction between workflow and MPI scheduling, are needed to separate concerns in real programs.

EMAN showed that scheduling workflow applications for efficient execution on the Grid requires relatively accurate estimates of node performance. Improvements in these estimates seem to help both advanced schedulers (such as those developed for GrADS and VGrADS) and straightforward ones (such as providing weights for randomized methods).

EOL: EOL led to much the same findings as EMAN - the need for homogeneous clusters and heterogeneous bags of resources. It also demonstrated the need for explicit handling of data movement in the VG. This handling includes both performing the movement itself and estimating its cost, which is needed for effective scheduling.

LEAD: LEAD has requirements much like EMAN and EOL for running an ensemble of simulations - the need for tightly connected clusters and loosely-coordinated larger collections. It adds the need for streaming data, as opposed to the fixed data collections used in the other two applications. LEAD also adds a more explicit requirement for fault tolerance, as well as some interesting (and not yet fully implemented) options for trading fault tolerance for output fidelity.

GridSAT: GridSAT has very different requirements from the other applications. Here, the set of tasks is extremely dynamic, thus pointing out the requirement for run-time expansion (and contraction) of the VG. It also relies more heavily than the other applications on topology information (in particular, identifying resources with good connectivity). Finally, fault tolerance is useful in GridSAT both to handle unreliable resources and to effect rescheduling when new resources become available.

2 VGrADS Programming Tools (Rice, UH, USC)

Performance Models: We have demonstrated that tools that automatically construct performance models can be developed and that the constructed models are accurate enough to be used in scheduling applications on the Grid. Furthermore, we have demonstrated that merging static and dynamic analysis of application binaries can produce *processor-neutral* predictive models of cache behavior and instruction execution that are surprisingly accurate (usually within 10%).

Workflow Scheduling: Through our application studies, we have demonstrated that the VGrADS strategy of using performance models for scheduling, particularly DAG scheduling, is extremely effective on heterogeneous computational resources, improving performance over current performance-blind strategies by up to factors of 20 in some cases.

3 VGrADS Execution System (UCSD, UCSB, UTK, UNC)

vgDL Design: Based on early evaluations, the vgDL language strikes a good balance between simplicity and flexibility for many applications. Application case studies, which are representative of popular grid application scenarios, have demonstrated that simple VG descriptions are rich enough to capture the critical resource needs of applications in a way that allows for efficient application deployment and optimization.

vgES Architecture and Implementation: Initial experiments with internal software releases demonstrate that the system, whose architecture integrates extant Grid services, enables straightforward and efficient deployment for a range of grid applications. Results presented in our CCGrid'05 research article show that vgFAB performs fast and high-quality resource selection and binding even in shared resource environments with high levels of resource contention.

VG Abstraction: While we have demonstrated that the Virtual Grid (VG) abstraction is effective for the purpose of resource selection, the VG is also a powerful runtime abstraction. Once instantiated, the VG is a living entity with which the application can interact dynamically. This interaction, as embodied in the vgES APIs, can consist of tracking dynamic information about the resources, evolving the VG by adding or removing resources, dynamically managing which application component runs on which portion of the VG, being notified of resource failures, etc. This in turn provides a set of simple and extensible mechanisms that allow dynamic execution adaptation both for performance and for fault-tolerance. This powerful abstraction is much richer that those used in current Grid application deployment software and should be attractive for such software. And indeed, we have already found clear traction between our work and the Pegasus environment (part of the widely-used Virtual Data Toolkit developed by the Globus team at USC), for which the use of the VG abstraction is currently being prototyped.

Automatic Resource Characterization: Numerical resource characterization through automatic modeling techniques can produce statistically rigorous predictions of future performance. In particular, it is possible to predict resource availability and batch-queue waiting times with verifiable confidence bounds.

The scale of the VG performance monitoring and forecasting is enhanced through abstraction and virtualization. By analyzing the statistical properties of different resources, the monitoring system can automatically present a virtual "picture" of the grid resource pool in terms of quantitative measures.

By making the statistical analysis both high-performance and portable, quantitative VG virtualizations can be delivered with high performance and scalability since only a small fraction of the measurement data must be transferred and stored.

III. VGrADS Education, Outreach, and Training Activities

The following sections describe VGrADS Education, Outreach, and Training (EOT) activities during the current reporting period (6/1/04-5/31/05)

1 Training and Development Activities

Much of VGrADS training effort has gone toward training and development at the college, post-graduate, and professional levels.

1.1 Inter-institutional Collaboration

The VGrADS project has provided opportunities for graduate students to become involved in an exciting and important research project.□Through participation in VGrADS project meetings, email, and phone conversations, students have been able to interact with, learn from, and contribute toward the research of off-site VGrADS participants.□The multi-site nature of this project has given students first-hand exposure to a wider range of research approaches and specialty areas than would typically be possible. Notably, this includes discussion and collaboration with several world-renowned researchers from other institutions with whom they would not normally interact. These students bring their insights back to other students in their research groups who are not exposed to as many "outside" collaborators, enriching the experience for other graduate students as well.

1.2 Student Exchanges

Graduate students working on the VGrADS project have the opportunity to broaden their educational and research experience and to strengthen their collaborations with colleagues at other VGrADS institutions by visiting other VGrADS sites for extended periods of time. For example, Ryan Zhang, a first-year student at Rice University, will spend the summer of 2005 at the University of California San Diego (UCSD). He will engage in research with VGrADS personnel at UCSD, primarily aimed at more closely linking the programming tools group with the execution system group. Detailed arrangements for this visit, such as locating housing, are already underway. Rice will continue to pay Ryan Zhang's salary and expenses while he is at UCSD; UCSD will handle all local arrangements details and the UCSD administrative paperwork. Preliminary plans are also underway for Dan Nurmi (UCSB) and Gurmeet Singh (USC) to make extended visits to Rice University this summer.

1.3 Distributed Software Engineering

The VGrADS project has been a vehicle for collaboration among faculty, students, and staff.□Additionally, the VGrADS project has provided students with a chance to build a very large-scale system in which all of the components must work together efficiently. The students have learned goal-setting and management techniques for distributed teams, and have learned how to use a variety of group communication techniques to make distributed teams effective. Since research groups are developing components of the system at various VGrADS sites, the project has also provided an opportunity for participants to collaborate closely with researchers with different expertise.

1.4 Courses

With support from their institutions, several VGrADS PIs have developed and taught courses that cover Grid technologies and other aspects of high performance distributed computing. These courses provide training for graduate students interested in undertaking Grid research projects. The following two subsections describe courses taught by Rich Wolski at the University of California, Santa Barbara (UCSB) and Andrew Chien at the University of California, San Diego (UCSD), respectively.

1.4.1 [UCSB] CS290I: Grid Computing

Tools developed as part of the VGrADS project and several of the GrADSoft tools developed as part of the NSF GrADS project have been used as the basis of a project-oriented graduate course in Computational Grid computing at UCSB. In it, students study the dynamics of performance fluctuations; design applications and schedulers based on this analysis, and implement their solutions using Grid tools. Specifically, the students:

1. experiment with the Grid technologies that are currently most popular,
2. build Computational Grid programs using these technologies, and
3. evaluate their effectiveness, particularly with respect to their performance.

Thus, VGrADS and GrADS software tools have enabled new educational experiences for graduate students at UCSB.

1.4.2 [UCSD] CSE225: Grids and High Performance Distributed Computing
http://www-csag.ucsd.edu/teaching/cse225s05/

Computational and Data Grids are open, shared resource infrastructures that support high-capability dynamic distributed applications. These infrastructures are the focus of national research projects in many countries (USA, UK, Japan, Korea, India, etc.), and the commercial strategies of major computing companies (IBM, Sun, Hewlett-Packard, etc.). For Grids to realize their potential, robust solutions to a number of difficult problems must be developed.

CSE225 consists of an overview of the unique attributes of grids followed by an examination of model problems for these challenges. These model problems include:

1. Dynamic Applications are Resource Aware: "Beauty is in the eye of the beholder."
2. Open Resource Sharing: "You can't always get what you want."
3. Federated Security: "If you can't trust the authorities, who can you trust?"
4. Dynamic Distribution of Data-Intensive Applications: "Let the data decide."

Students who complete CSE225 are well prepared to undertake research in Grids and high performance distributed computing.

2 Outreach Activities

In addition to the usual VGrADS teleconferences and face-to-face meetings, the Education, Outreach, and Training (EOT) group at Rice held several meetings to develop and coordinate outreach activities, with particular emphasis on outreach activities for the summer of 2005 and beyond. The following sections describe VGrADS outreach activities during the reporting period and near-term planned VGrADS outreach activities.

2.1 Collaboration with Alliances for Graduate Education and the Professoriate (AGEP)

AGEP (http://rgs.rice.edu/grad/agep/index.cfm) is a program of the NSF EHR directorate. One goal of Rice's AGEP program is to provide a year-round community experience for graduate students from under-represented groups. The program encourages Science/Math/Engineering (SME) graduate enrollment, supports Ph.D. degree completion, and exposes the students to rigorous academic discipline. Together with strong commitments by Rice University, this program is designed to permanently alter the institution's graduate student diversity. The AGEP summer program at Rice provides hands-on research experience to undergraduate students in SME disciplines, with an eye toward giving the students a solid foundation for the remainder of their undergraduate course work, developing professional relationships, and gaining a sense of what graduate school will be like, particularly at Rice University. VGrADS leverages this program to provide an opportunity for outreach to under-represented groups by participating in the AGEP summer program. AGEP leverages VGrADS to reach more students, through our direct funding of additional participants.

The intended milestone for year one of VGrADS was that we would host the first VGrADS AGEP participants at Rice University. Due to a late start in recruiting, no VGrADS students were found for AGEP the first year. We are rolling the funds that would have been used for these students in year 1 forward, and expect to support additional students in years 2 and 3 (three students each year, rather than two as planned). Since we could not meet the intended milestone of sending AGEP students to the Grace Hopper Conference, we decided that the aims of diversity could be met by supporting Rice female CS students. (Details are provided in the next section.)

Recruiting for the summer of 2005 is proceeding better than recruiting for last summer. Approximately 110 students have applied to the AGEP program, of whom more than 20 have specifically expressed interest in working with VGrADS. At this writing we have accepted three undergraduate minority women as VGrADS-supported AGEP students this summer, and are awaiting their formal acceptances. (If one or more cannot accept, there are other viable candidates as well.) Each will have a research project, typically running an experiment on the VGrADS infrastructure and analyzing the results. Several possible VGrADS projects for summer students have also been discussed, some related to the evaluation of scheduling methods, some concerning launching Grid applications, and others related to the analysis of mapping resources for the VGrADS scheduler. These projects could lead to conference and journal publications (on which the student would be a co-author), and information gathered from the projects could lead to the design of improved schedulers for VGrADS. Moreover, we hope that the research projects themselves motivate the students to continue into graduate school, as some former AGEP students have done.

2.2 Participation in Conferences Focused on Diversity in Computer Science

As planned, VGrADS outreach-based conference participation has focused on supporting activities at the Grace Hopper Celebration of Women in Computing and at the Richard Tapia Celebration of Diversity in Computing. Both meetings are devoted to increasing diversity in computer and computational science, and were chosen for that reason.

In October 2004, Rice's Center for Excellence and Equity in Education (CEEE) presented a panel at the Grace Hopper Celebration of Women in Computing. CEEE employee Michael Sirois chaired the panel, which shared methodologies that have been successful in encouraging diversity in science, technology, engineering, and math (STEM) programs. Rice CS student Christy Beatty and CEEE workshop teachers also participated in the presentation. Christy Beatty and Elspeth Simpson, another Rice CS student, also conducted their own "birds of a feather" (BOF) session on problems associated with achieving gender equity in CS at the university level. VGrADS provided travel support and also leveraged support through CEEE, Rice's Dean of

Engineering, and the Computer and Information Technology Institute (CITI) at Rice to send six students and five adult participants and attendees.

Plans for FY06 include participation in the Richard Tapia Celebration of Diversity in Computing 2005, which will be held in Albuquerque, New Mexico, October 19-22, 2005. Keith Cooper, VGrADS PI, will give a keynote speech at the conference. We also anticipate presenting a panel on Grid computing. Charles Koelbel, one of the VGrADS PIs, has agreed to organize and chair the panel, and most of the panelists will come from VGrADS institutions (including one of the Rice AGEP summer students). In addition, we plan to provide and leverage funding to ensure a high level of student attendance at the conference.

2.3 Participation in NSF-funded Computer Science Computer and Mentoring Partnership

As planned, Richard Tapia and Keith Cooper, PI's of the NSF-funded Computer Science Computer and Mentoring Partnership (CS-CAMP) project (http://ceee.rice.edu/cs-camp/), both made presentations at CS-CAMP sessions (Summer 2004) for high-school CS teachers and high-school girls. Cooper and Tapia provided insights regarding careers in computer science, as well as diversity-related issues STEM careers. They will be presenting again at the next CS-CAMP (Summer 2005).

We hope to extend this successful program beyond Rice if additional support can be obtained. Toward this end, we are preparing a proposal to the NSF Broadening Participation in Computing (BPC) program to replicate CS-CAMP at UCSD or USC.

We plan to disseminate materials developed by the program through the newly-announced EPIC network. This organization is the successor to the NPACI/EOT program. Other VGrADS partners will integrate the VGrADS materials into their existing programs, and the materials will also be distributed nationwide through forums such as the National Computational Science Institute and the annual Supercomputing conference.

2.3.1 Seeking External Funds to Increase Diversity

VGrADS PI Keith Cooper is also the PI on a grant from the Luce Foundation to fund a pair of Clare Booth Luce Fellowships in Computer Science at Rice. The intent of this program is both to increase the pool of female applicants to the Department's Ph.D. program and to provide a generous stipend ($25,000) to two entering female Ph.D. students. Two fellowship offers were extended this year; the results are pending.

2.4 Computer Science Community Interactions

VGrADS researchers have presented (and will continue to present) VGrADS research results and ideas in a variety of forums, including technical computer science presentations, presentations to applications groups, and presentations to students.

The VGrADS project had a variety of professional outreach activities at the annual SC2004 Conference (Pittsburgh, PA, November 6-12, 2004).□Other sections of this report detail the research advances that were reported there, including paper and poster presentations.□We also gave a number of talks with accompanying demonstrations in exhibit booths, including:

- Ken Kennedy (Rice) teaming up with Rich Wolski, and Wahid Chrabakh (UCSB) to describe VGrADS and demonstrate the GridSAT application at the SDSC and Rice research booths.
- Charles Koelbel, Ken Kennedy, Anirban Mandal (all Rice), and Lennart Johnsson and Bo Liu (both UH) - in various combinations - giving overviews of VGrADS and demonstrating the EMAN application at the Argonne, Rice, and UH booths.
- Jack Dongarra (UTK) speaking about fault tolerance methods developed under VGrADS at the Rice booth.

Most of these demos attracted reasonable audiences, and represented good outreach to the high-performance computing and research communities by the project. Also, Charles Koelbel (Rice) presented VGrADS plans and preliminary results at a Birds-of-a-Feather (BOF) session on "How do we develop, debug, and tune applications on Grids?" This was a particularly important contact point with the wider Grid community, since a Global Grid Forum research group organized the BOF. Less VGrADS-specific, but higher-profile, were the participation of several PIs in two extremely popular panels:

- Dan Reed (UNC) was the moderator and main speaker of "National Priorities for Computational Science - A PITAC Town Hall Meeting". This session gathered input for the President's Information Technology Advisory Committee, which advises the White House and Office of Management & Budget.
- Charles Koelbel (Rice) and Jack Dongarra (UTK) were panelists on the "Future of Supercomputing". This session reported the results of the recent Computer Science and Telecommunications Board study of supercomputing, which was commissioned by the Department of Energy in response to inquiries from Congress.

Both panel sessions were overfull due to audience interest in federal government policy.□Although not traditional outreach, bringing Grid research into these arenas contributed substantially to the broader impact of VGrADS.

IV. VGrADS Contributions

1. Contributions within Discipline

VGrADS activities and findings during the funding period, which are described in more detail in the "Activities" and "Findings" sections of this report, included research results and associated implementations that will ultimately contribute toward computer science research, particularly in the area of distributed, heterogeneous computing. Research highlights from the VGrADS project during the funding period include:

- VGrADS researchers developed new methods for scheduling workflow applications (those with multiple components linked by data and control dependence) on distributed computational grids. This will allow much better performance for many scientific applications, in fields ranging from bioimaging to climate modeling to genome mapping. Our workflow scheduler differs from others in wide use (e.g. Condor's DAGMAN system) by using information about later tasks to optimize both computation and communication performance simultaneously.

  Specifically, the system schedules components as a task graph. The computation cost for each component task is estimated by a hybrid performance model, composed of a hand-generated cost of floating-point computations (parameterized by type of processor) and an automated estimate of memory hierarchy costs (parameterized by cache sizes and other system characteristics). The communication cost between component tasks is estimated from latency and bandwidth information derived from NWS. We then apply heuristics (MIN-MIN, MIN-MAX, and Sufferage) to choose the best mapping of components to nodes.

  Recent experiments have demonstrated the scalability of this scheduler to multiple heterogeneous clusters. Our recent (albeit preliminary) experiments with the EMAN electron microscopy image package show that it gives up to 120% better performance than simple random scheduling, and 50% better than a more sophisticated (but still randomized) scheduling.

- To support measurement of failure indicators, a prerequisite for predicting failures and providing reliable virtual Girds, we integrated the extant University of North Carolina's Health Application Programming Interface (HAPI) toolkit with the UCSB Network Weather Service (NWS). The HAPI package is used discovery and monitoring of health-related diagnostic information on Linux clusters. By leveraging the open source, flexible software offered by both HAPI and NWS, we have extended the state-of-the-art in aggregate cluster health monitoring.□Whereas NWS is focused on a high-level data model to derive forecasting models, HAPI is focused on ease of data acquisition, extending the breadth and depth of raw data available for downstream use.□

- Extending our previous work from the GrADS project on quantitative performance contracts, we are developing a qualitative temporal performance validation approach for virtual grids. Our goal is to reason about temporal events to differentiate between severe, persistent behavioral violations and transient ones. Our framework has three major components: (a) classification of temporal behaviors observed in scientific workloads, (b) development of a temporal algebra for temporal behaviors and interactions, and (c) a methodology for predicting behavior interactions

  We are collecting system-level, quantitative resource usage data and are analyzing the data using statistical and clustering techniques. The results define a set of high-level, qualitative behavioral phases based on resource type (e.g.: CPU-intensive, IO-intensive, and CPU+IO intensive) and resource usage pattern (e.g.: oscillatory or steady). We have developed a temporal algebra that expresses phase behaviors, along with a methodology for predicting the phases that result from the convolutions of phases.

- To support fault-tolerant prediction, we have worked out a method for both modeling and predicting availability times, including the duration that a machine will run until it restarts (availability duration), for Grid resources. We do this by fitting live performance data to standard parametric models (Weibull, 2-phase hyperexponential, and 3-phase hyperexponential) using a maximum likelihood estimate (MLE) approach. Our results indicate that this enables us to predict availability duration with quantifiable confidence bounds and that these bounds can be used as conservative bounds on lifetime predictions.
  Based on this work, we have also developed a system for automatically determining an "optimal" checkpoint schedule for an application. The word "optimal" is in quotation marks because it relies on the parametric availability model (which is chosen above), making the optimality depend on the goodness of the model fit. In practice, however, the model fits are sufficiently good on a variety of systems. Work continues to validate the quality of models, to empirically evaluate the checkpoint efficiency (difficult because the failure times are so far apart), and to extend this work to use non-parametric statistics.

- Large-scale network simulation is an important technique for studying the dynamic behavior of Grid applications, as well as other aspects of networks. Simulating Grid applications requires both large scale, which in turn requires advanced strategies for load-balancing the simulation itself, and realism, which in turn requires detailed models within the simulation.

To meet the former need, we developed a new hierarchical profile-based load balance technique (HPROF) for our MicroGrid system. The key idea is to cluster network nodes and explicitly control the tradeoff between simulation efficiency and available parallelism. The effect is to produce robust and superior performance for large-scale networks (20,000 simulated routers), improving load balance by 40% and simulation time by 50% on our 128-node cluster.

To meet part of the latter need, we developed new methods for generating large network topologies. The basis of our methods is to apply publicly known BGP policies in the process of creating topologies, thus generating realistic routing of traffic in the topology. This is a significant difference from previous approaches, which typically generated a realistic physical topology but (unrealistically) assumed flat shortest-path routing.

Additionally, the VGrADS project has supported the development and/or enhancement of software components and products. For example:

- NWS Scalable Performance Virtualizer: VGrADS researchers have developed a scalable performance virtualizer for the NWS. The purpose of this software component is to present a virtualized view of the grid resource performance characteristics to the virtual grid execution system (vgES). Using this new NWS facility, the vgES will be able to determine a virtual performance topology of the grid resources at hand in a way that scales with the number of resource characteristics (and resources) to be monitored. Scalability is gained through this virtualization since the NWS does not need to monitor all resources at all time. This software component is currently a prototype. The software component will be included in a future standard NWS release (which will be available from the NWS web site at UCSB) and, subsequently, in a release of the NMI software tools.

- HAPI/NWS Integration: VGrADS researchers integrated the University of North Carolina's Health Application Programming Interface (HAPI) toolkit with the UCSB Network Weather Service (NWS). The HAPI package, developed with DOE funding, consists of a C code library, sample applications, and documentation, and is used for discovery and monitoring of health-related diagnostic information on Linux clusters. Such information typically consists of system temperatures, voltages from the power supplies, disk drive error rates, fan speeds and system load. HAPI collects this data through ACPI, SMART, lm_sensors, and IPMI interfaces in a flexible, extensible plugin model.□

  By leveraging the open source, flexible software offered by both HAPI and NWS, we have extended the state-of-the-art in aggregate cluster health monitoring.□Whereas NWS is focused on a high-level data model to derive forecasting models, HAPI is focused on ease of data acquisition, extending the breadth and depth of raw data available for downstream use.□We continue to work with NWS authors to include the HAPI integration with future NWS distributions as part of the National Middleware Initiative (NWI). In addition, we plan to distribute HAPI separately.

2) Contributions to Other Disciplines

As indicated in the VGrADS highlights listed under "Contributions within Discipline," many of the ideas and associated implementations developed under the VGrADS project are relevant to application researchers interested in or currently using grid computing. The VGrADS project has also supported the development and/or enhancement of software packages that are used by a variety of application groups, including those application groups directly collaborating with VGrADS researchers.

3) Contributions to Human Resources Development

The VGrADS project has provided computer science research opportunities for graduate students and postdoctoral associates, including individuals from underrepresented groups. Through participation in VGrADS project meetings, email, and phone conversations, students and postdoctoral associates have been able to interact with, learn from, and contribute toward the research of off-site VGrADS participants.□ The multi-site nature of this project has exposed participants first-hand to a wider range of research approaches and specialty areas than would typically be possible. Additionally, graduate students working on the VGrADS project have the opportunity to broaden their educational and research experience and to strengthen their collaborations with colleagues at other VGrADS sites by visiting other VGrADS institutions for extended periods of time.

With support from their institutions, several VGrADS PIs are involved in teaching Grid-related courses. In one project-oriented graduate course in Computational Grid computing, VGrADS software tools, along with performance data and other software tools developed under previous funding, have been used as the basis of the course. In the course, students study the dynamics of performance fluctuations, design applications, and schedulers based on this analysis, and implement their solutions using Grid tools.

VGrADS researchers and staff have been actively involved in efforts to encourage high-school students, undergraduates, and graduate students from underrepresented groups to pursue careers in science, math, and technology fields. The programs and activities for students from underrepresented groups, which are described in more detail under "Outreach Activities," have included summer research experiences for undergraduates; mentoring programs for graduate students, undergraduates, and high-school students;

seeking funding for fellowships to increase diversity, and undergraduate participation in conferences devoted to increasing diversity in computer and computational science.

VGrADS researchers have presented VGrADS research results and ideas in a variety of forums, including technical computer science presentations, presentations to applications groups, and presentations to students. In particular, the VGrADS project was involved in a variety of outreach activities at the SC2004 conference in Pittsburgh, PA (November 6-12, 2004). VGrADS activities at SC2004 are discussed under both “Outreach Activities” and “Project Activities.”