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 (6/1/05 – 5/31/06), VGrADS research focused on the three inter-institutional efforts described in the following sections: *Applications (Section 1), VGrADS Programming Tools (Section 2),* and *VGrADS Execution System (Section 3).* 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.

During the funding period, the PIs also restructured the management of the VGrADS project and revamped the VGrADS milestones (Section 4) to reflect personnel changes in the project. The most notable change is Andrew Chien's departure to become Chief Scientist for Intel. As a result of Andrew Chien's departure, the VGrADS Execution System thrust is now under the leadership of Carl Kesselman. This has also resulted in some rebudgeting to increase support for Kesselman, as we have reported elsewhere.

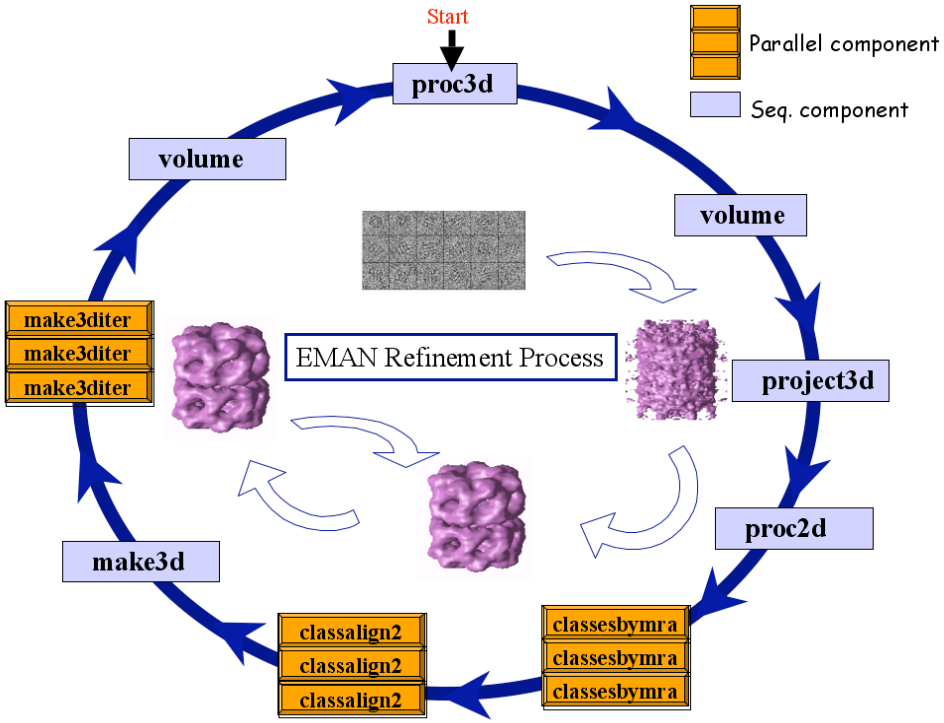
VGrADS includes researchers from Rice University; University of California, San Diego (UCSD); University of California, Santa Barbara (UCSB); University of Houston (UH); University of North Carolina (UNC); University of Southern California / Information Sciences Institute (USC); and University of Tennessee, 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, two PI teleconferences, one PI meeting, VGrADS planning workshops at UCSD (9/12-13/05) and Rice (2/23-24/06), one VGrADS-LEAD research workshop at UNC (4/18-23/06), and communication via VGrADS mailing lists. The fall 2006 VGrADS planning workshop will be held at UCSB (9/6-7/06). This summer, another research workshop will be held at USC/ISI (7/25-27/06). Research subproject participants also met on a regular basis to exchange ideas and develop research plans; the EOT group at Rice met five times to develop and coordinate education and outreach activities.

1 Applications (Rice, UCSD, UCSB, UH, UTK, 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, these applications 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, UH and UCSB 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 Dr. 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.

In our previous report, we have described how we used the scheduler to map EMAN computations effectively onto resources in a large Grid consisting of two Itanium clusters and one Opteron cluster across Rice and UH. That experiment showed that both accurate performance models and scheduling heuristics were effective in scheduling EMAN computations across multiple clusters. The level of accuracy (of performance models) 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).

However, we conducted these experiments with the assumption that the clusters were unloaded and were available instantly for EMAN computations. But, this assumption may not hold for non-dedicated clusters typically managed by separate batch-queue schedulers that implement local space-sharing policies. The time for a job in queue can be highly variable and potentially long – longer in many cases than the execution time itself. Hence, considering queue wait times while scheduling is absolutely essential. So, we equipped the scheduler to use predictions of when resources become available (using novel NWS methodologies developed at UCSB) along with accurate performance models to decrease the overall turnaround time of the workflow when executed on non-dedicated systems. We used the batch-queue prediction enhanced scheduler to run EMAN computations across five different clusters – three Itanium TeraGrid clusters in San Diego, Urbana-Champaign, and Chicago; one Itanium cluster at Rice; and one Xeon cluster at UCSB. The results showed that using batch-queue wait time predictions significantly reduces the turnaround time of EMAN computations.

The Grid scheduler produces a "plan" of workflow task to resource mappings describing when and where to execute individual tasks. We feed the "plan" to an execution runtime manager, which is a simple program that automatically creates job-submission files for the various tasks in the plan and submits them to the site controlling the chosen individual resources. The jobs then wait in the site's batch queue until they start execution (and a notice is sent back to the execution manager). When the jobs finish execution, another notice is sent back to the execution manager, which triggers the execution of the next workflow step.

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.

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 Python compilation project mentioned in Section 2.

1.2 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. 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.

1.3 LEAD

VGrADS is closely collaborating with another NSF funded ITR project, LEAD (Linked Environments for Atmospheric Discovery). The goal of LEAD is to build a scalable web services workflow infrastructure for meteorological data and models. The meteorology and the web services components of LEAD are being developed under a separate ITR award (NSF 0315594); the VGrADS team is collaborating with LEAD to apply resource selection, scheduling and provisioning techniques from the VGrADS research effort to the LEAD workflow.

The unique characteristics of LEAD lie in the dynamic workflow orchestration and data management, which allow 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. The definition of workflow in LEAD is subtly different from the definition in EMAN. 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 each workflow step is managed by a persistent service whereas EMAN’s tasks are not persistent. In addition, there is a need for

managing streaming data, as opposed to the fixed data collections used in the other applications.

Project members from both teams met at the Research Triangle Park in November 2005 to spearhead the collaboration. The team discussed short-term and long-term plans of collaboration and integration, including the system architecture, scheduling for both static and dynamic LEAD workflows and fault tolerance strategies for LEAD workflows. The LEAD-VGrADS integration is currently focused on designing the interfaces that will allow interaction between these two paradigms for a static LEAD workflow. An integration meeting was held in April 2006 where members refined plans and activities to compose software ideas and approaches.

During the reporting period, our focus was on specifically enhancing the functionality of the various VGrADS components that will interact with the LEAD architecture. We are building a performance model of the entire LEAD workflow that is able to approximately estimate data sizes and running times of all the stages in the workflow. The performance model will then be used in conjunction with Rice's scheduler and the UCSB's batch queue prediction in the resource selection mechanism of vgES. This interaction will be coordinated through a resource provisioning service. The resource provisioning service will be a web service in the LEAD framework.

1.4 FT-LA: Fault Tolerant Linear Algebra

UTK is exploring scalable techniques to tolerate a small number of process failures in large scale computing. The goal is to develop scalable fault tolerance techniques to help to make future high performance computing applications self-adaptive and fault survivable. The fundamental challenge in this research is scalability. To approach this challenge, we are (1) extending existing diskless checkpointing techniques to enable them to better scale in large scale high performance computing systems; (2) designing checkpoint-free fault tolerance techniques for linear algebra computations to survive process failures without checkpoint or rollback recovery; (3) developing coding approaches and novel erasure correcting codes to help applications to survive multiple simultaneous process failures. The fault tolerance schemes we are developing are scalable in the sense that the overhead to tolerate a failure of a fixed number of processes does not increase as the number of total processes in a parallel system increases.

Various high performance distributed linear algebra routines are being examined to determine how to best support process fault tolerance. FT-MPI (a fault-tolerant implementation of MPI 1.2 developed at UTK) is used for communications, and recovery techniques are incorporated into the libraries to allow them to recover from errors in a fast and scalable fashion.

2 VGrADS Programming Tools (Rice, UCSB, UCSD, UH, UTK)

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. We have previously reported on our development, in conjunction with the systems groups, of appropriate abstractions to simplify this task. As interfaces to the Virtual Grid Execution System (vgES) 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:

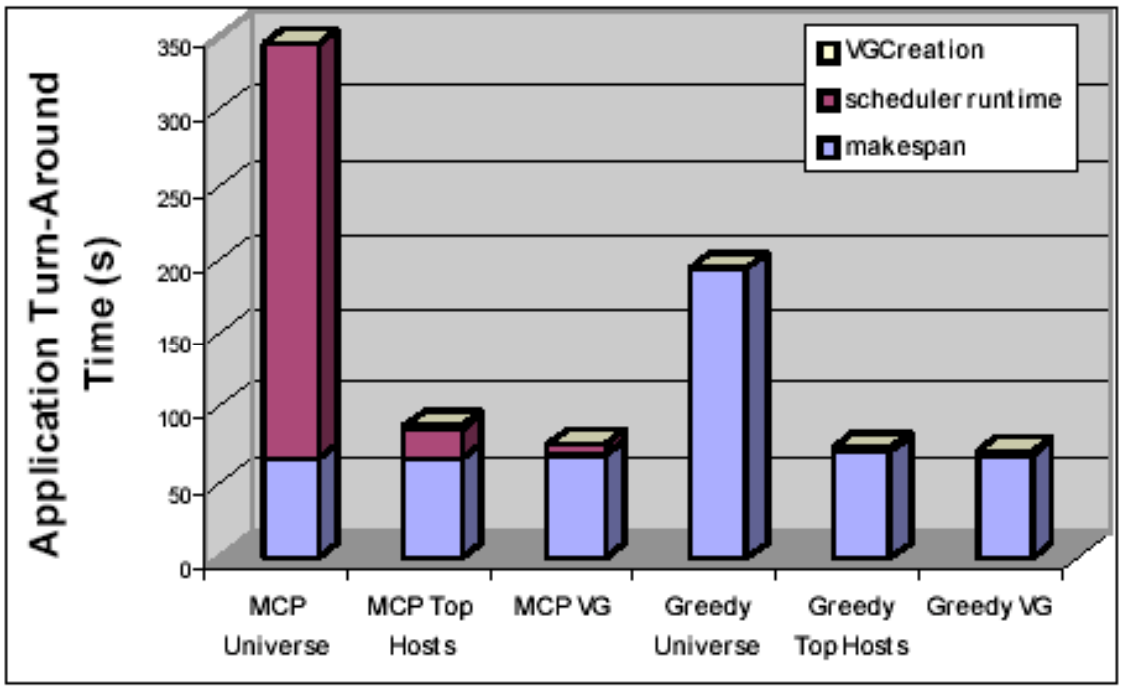
1. Improved scheduling for workflow computations on the Grid and VGs,
2. Performance prediction of application components to be mapped onto the Grid,
3. Compiling and optimizing node programs for use in a Grid environment,
4. Generating Service Level Agreements (SLAs) for applications,
5. Fault-tolerant libraries for MPI and OpenMP, and
6. Construction of Grid workflows from high-level scripts.

The following subsections discuss each thrust 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 Applications on Virtual Grids

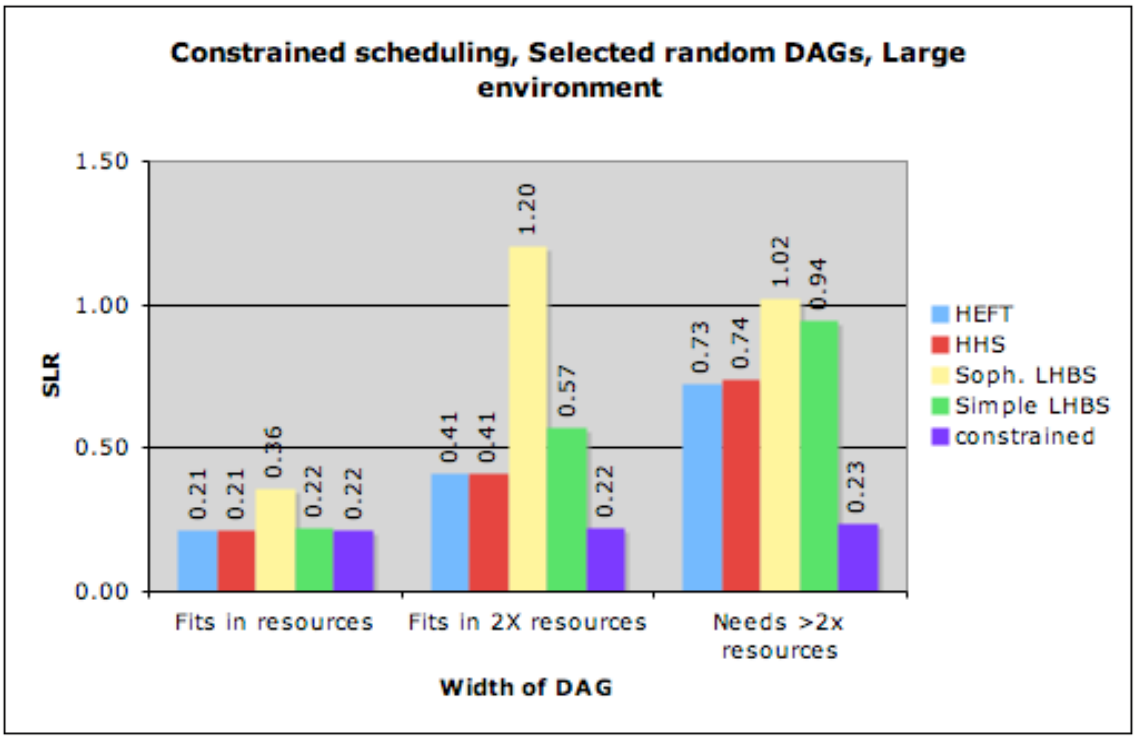
As we have previously reported, we have done a great deal of development of scheduling strategies for the EMAN and EOL applications. This year, we have consolidated and extended that work to develop new scheduling methods (e.g. the two-phase scheduling described in the next paragraph) and to explore new scheduling scenarios (e.g. scheduling for nondeterministic resources and for batch queued systems).

Researchers at Rice and UCSD have studied extensions to existing schedulers to make full use of Virtual Grid (VG) technology. We refer to this as the "two-phase" scheduling strategy, where the first strategy is selecting a VG, and the second phase is mapping the application onto the chosen VG. The UCSD group reported their findings at IPDPS, where they found that a simple greedy mapping heuristic coupled with our VG abstraction produced excellent application turnaround times (defined as total time for scheduling and the application itself). We reproduce the following typical graph from that paper. Note how the bars marked "VG" are consistently competitive in makespan (computation time) and low in scheduling time (the sum of VG creation and scheduler).



The Rice results have not (yet) been accepted for publication.

In a related vein, Rice researchers have embarked on a comparative study of list-based and level-based schedulers, exemplified by the Heterogeneous Earliest Finish Time (HEFT) and Levelized Heuristic Based Scheduling (LHBS) strategies. Although we have only recently submitted this work for publication, we think the results are striking. In short, they show that simple scheduling, particularly in conjunction with reducing the number of resources considered, is highly effective for scheduling. In particular, we now understand better how constraining the choice of resources improves both scheduler performance *and the generated schedule*, by avoiding "false economies" in trading computation performance for communication performance.



Researchers at UTK are also extending scheduling concepts to deal explicitly with dynamic environments. Most workflow scheduling algorithms are based on the *deterministic* model, driven by (assumed) accurate knowledge of the tasks and their performance. Although a schedule may be optimized with respect to these assumptions, it may perform poorly in real systems owing to run-time variations.

Dealing with nondeterministic environments is more challenging than static scheduling. In order to increase the robustness of a schedule, one possibility is to judiciously overestimate the execution time of each task according to its variability, hoping that the real execution time will not exceed the estimated one. It is obvious that this approach could significantly reduce the resource utilization. In job shop scheduling, slack is widely used to increase the robustness of schedules. The slack of a task represents a

time window within which it can be started without extending the makespan and it is intuitively related to the robustness of the schedule. Larger slack tends to absorb the task execution time variance with little delay.

We first demonstrate the usefulness of slack in absorbing uncertainty in workflow scheduling. Using slack to represent the robustness of a schedule, we develop a genetic-algorithm-based heuristic to find schedules more robust to the non-deterministic nature of the task execution time. Genetic-algorithm-based task scheduling algorithms normally take the makespan as their objective function. In order to take into account both the robustness and makespan of a schedule, we include both makespan and slack in the objective function. Unfortunately, slack and makespan are two conflicting factors of the problem. Optimizing only makespan will result in schedules with small slack, which are thus less robust to task execution time variability. Conversely, optimizing slack alone tends to give schedules with good robustness but large makespan. To handle this multi-objective optimization problem, the ε-constraint method is employed. The scheduling algorithm tries to find the schedules with maximal slack without exceeding the specified makespan upper bound. Simulation and evaluation systems are developed to compare the performance of different schedules. Although the robustness of a schedule is a desirable property and is conceptually easy to perceive, it is difficult to measure quantitatively. There are several attempts to define it according to different perspectives of the problem. Two new measures of robustness were developed as part of this work.

Researchers at UCSB and Rice have collaborated on a prototype scheduler for Grids in which resources are controlled by batch queues. Although this research has just been submitted for publication, our results seem promising. The key insight for this work is that standard top-down schedulers, such as the ones Rice researchers have successfully used in the EMAN application, use an Estimated Start Time (EST) for each resource in choosing the next task to be mapped. In the original scheduler formulation, the EST for a task only depended on the completion times of its predecessors and the communication time to receive its data. We use the predictions of batch queue wait time derived by researchers at UCSB to refine this estimate. Our new EST for mapping a task to a resource is essentially the maximum of the old EST and the time when the resource will become available through the queue. Initial results with this estimate have been quite encouraging, as we demonstrated at the SC'05 conference. We have submitted a paper on the topic to SC'06.

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 distinct referenced memory blocks, between accesses to the same cache block. Results of this approach were published in SIGMETRICS 2004 and LACSI 2005, showing that, for a set of six scientific applications, this methodology usually produces estimates within 10% of the measured number of cache misses, for a large range of input data sizes, and for caches of different associativity levels and sizes.

In the last year, we have been working on rewriting and refining our instruction scheduler, which we use among other things to estimate the computation part of an application's execution cost, and to assess what factors limit the performance of an application on a target architecture. Our current efforts are aimed at designing and implementing a methodology to predict an application's bandwidth requirements at each level of the memory hierarchy assuming perfect prefetching, and to estimate the realizable bandwidths in the absence of prefetching. These new techniques will help produce a more detailed execution model, which in turn should result in more accurate estimates of execution time.

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. As detailed in previous reports, we have adopted a strategy of using LLVM (from Adve's group in Illinois) as a platform to enable compilation on heterogeneous processors. In effect, we perform Just-In-Time (JIT) compilation of the application when we assign it to a Grid node. Our efforts this

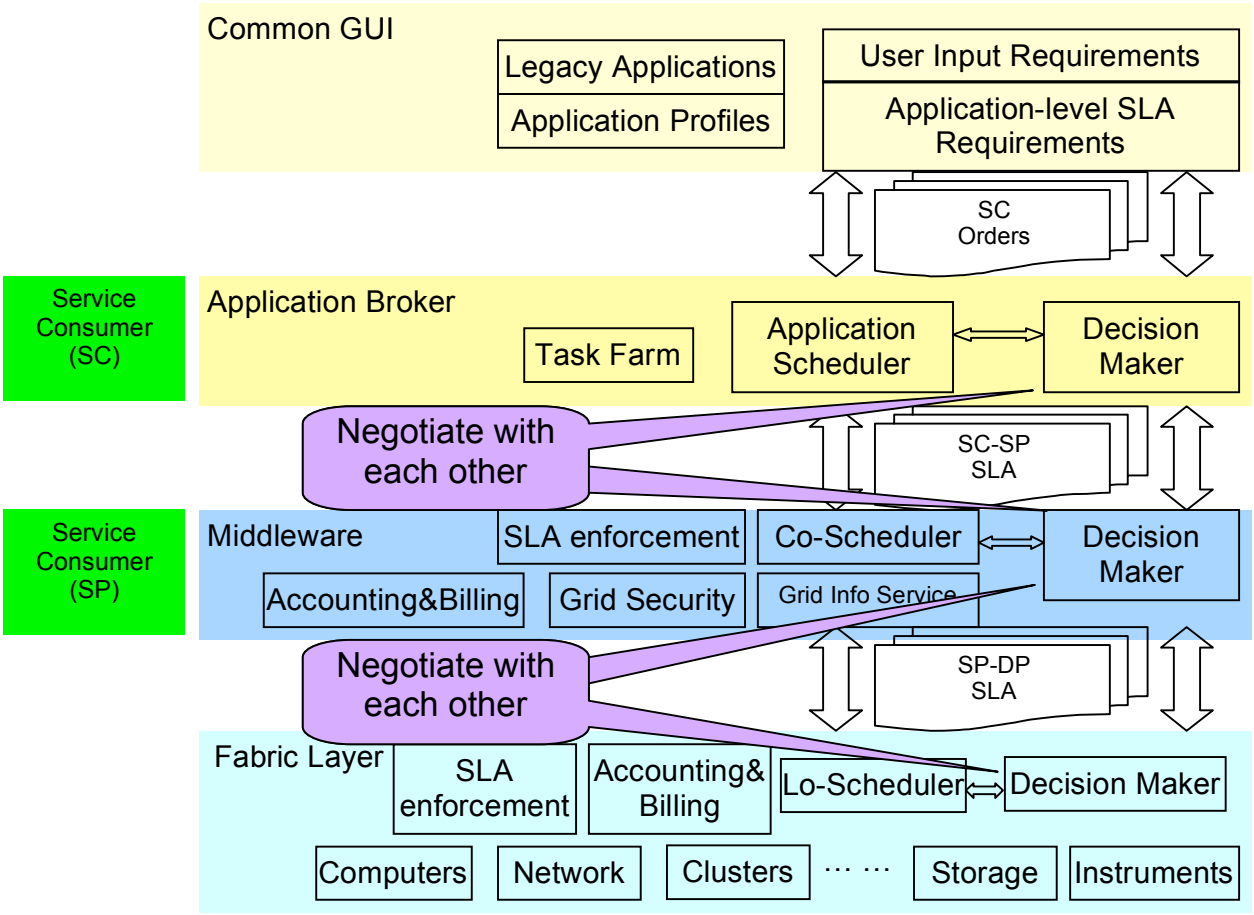
year have focused on studying the effect of just-in-time compilation on traditional compiler optimization. The most important outcome of this work was an International Symposium on Code Generation and Optimization paper that presented a novel graph-coloring register allocator tuned to JIT contexts. The allocator reaps most of the benefits of traditional (non-JIT) graph coloring register allocation, while avoiding their prohibitive compilation costs in a JIT environment.

2.4 Generating Service Level Agreements

Service level agreements (SLAs) are contracts between service providers and customers that define the services provided, the metrics associated with these services, acceptable and unacceptable service levels, liabilities on the part of the service provider and the customer, and actions to be taken in specific circumstances. They are the contractual component of QoS and are usually implemented as part of a larger Service Level Management (SLM) initiative. It is clear that for a collaborative computing environment like the Grid, SLA is of critical importance if the service providers and service consumers are to deliver services efficiently and cost effectively.

The SLA itself must be of sufficient detail and scope for the service covered. Typical SLA sections include: a) services to be delivered; b) performance measuring, monitoring and reporting; c) problem management; d) compensation such as price and service delivered times; e) customer duties and responsibilities; f) warranties and remedies; g) security issues; h) intellectual property rights and confidential information; i) legal compliance and resolution of disputes; j) termination; and k) signatures. Other sections may also be applicable. Clearly, setting up a digital SLA in not easy.

Besides drawing up an SLA, measuring and monitoring the services, periodic review, conducting and monitoring service improvement, and amending the SLA are normally included to complete the life cycle of SLM. The following figure depicts a layered architecture of the SLA-driven framework with the components required to implement SLM.



***Co-Scheduler**: coordinative scheduler*
***Lo-Scheduler**: local scheduler in comparison to Co-scheduler*
***SC-SP SLA**: SLA between service provider and service consumer*
***SP-DP SLA**: SLA between service provider and device provider. It is an SC-SP SLA between the service provider and the device provider that is transparent to the SC.*

To start the service delivery cycle, the first step is to implement the SLA. A consumer first looks up a service catalog and proposes a draft. Then the two parties negotiate, review the underlying contract, renegotiate as necessary, and finally settle on an agreement. The objective for the service consumer is usually to finish the tasks as soon as possible within a reasonable cost. The goal of the service providers is usually maximizing the overall profit with limited resources. From the above diagram, we can see that prediction of the service execution time and negotiation make up the core of setting up a digital SLA for this scenario. In large-scale distributed shared resources such as Grids, maximizing the autonomy of SLA implementation is key to making SLA execution practical.

We predict the service execution time with the parameterized performance models. For example, in deploying EMAN service, the models take the input scale and computing resources' attributes.

2.5 Fault tolerant MPI: FT-MPI / OpenMPI

Fault Tolerant MPI (FT-MPI) is a full 1.2 MPI specification implementation developed at UTK that provides process level fault tolerance at the MPI API level. FT-MPI is built upon the fault tolerant HARNESS runtime system and framework for distributed computing that is also being developed at UTK. FT-MPI survives the crash of n-1 processes in an n-process job, and, if required, can restart them. However, it is still the responsibility of the application to recover the data-structures and the data on the crashed processes.

FT-MPI provides a high performance implementation of MPI, resulting in performance that is similar to MPICH2 or LAM. The FT-MPI approach to fault tolerance allows the user to code their applications so that the over-head of the fault tolerance is minimized.

Active MPI development at UTK has moved to the OpenMPI project, which is combining technologies and resources from several other projects (FT-MPI, LA-MPI, LAM/MPI, and PACX-MPI) in order to build the best MPI library available. A completely new MPI-2 compliant implementation, OpenMPI offers advantages for system and software vendors, application developers, and computer science researchers.

2.6 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, ISI)

The latest research directions address the continuing development of the virtual grid execution system (vgES) to support complex, adaptive workflow applications. We are currently focused on three distinct areas. First, the previous version of the virtual grid had a crude method to allow vgDL requests to specify *when* resources should be made available; that version dealt exclusively with time-sharing or dedicated resources that were made immediately available to the user. To that end, we are refining the time notion in the vgDL language and developing a plug-in architecture to accommodate a variety of resource managers, including batch queues and lease-based systems. The key motivating application for this work is the LEAD workflow. Its requirements are indicative of the complicated machinations required for adaptive, real-time applications. Second, we have enhanced the execution system's ability to successfully find the resource set that matches the application's resource request. We do this by incorporating *binding*, the ability to perform useful work on found resources, as a first-class selection constraint. Finally, our research continues to address other stages of the application life cycle, beyond that of finding and binding federated resources. In particular, we have developed a tool to help applications manage and monitor data generated across the federated environment in a robust and accurate fashion.

We demonstrated the first generation virtual grid execution system, including finding/binding (vgFAB), remote execution facilities (vgLaunch), and monitoring (Mortar), at Supercomputing 2005, Seattle. This demonstration illustrated the power and flexibility of vgES and Mortar by building an adaptive application controller for a grid-enabled application, EMAN. The controller adapted application execution to changes in the resource environment and perceived application performance. Multiple members of the execution system team presented these demonstrations.

3.1 Improved Resource Selection through Integrated Finding and Binding

The common architecture of traditional resource brokering systems separates resource selection from resource binding. These systems focus on resource selection, providing sophisticated resource specification languages and resource selection algorithms. However, the separate approach cannot deal with binding failures efficiently. In the real world, resource binding may fail due to inaccurate resource information, authentication failure, and, most importantly, contention for popular resources. The key issue is that binding failures cannot be resolved until the system attempts to bind the resources after the resources have been selected.

We propose integrated selection and binding to solve the resource selection and binding problem. The key idea is to treat binding as a first-class constraint during resource selection. Several features, such as composition operators in our resource description language (vgDL) and our data representation (resource classification and database

schema), enable this approach. This combination allows us to efficiently allocate complex resource collections even in the presence of competition for resources. Our empirical evaluation shows that the integrated approach produces higher-quality solutions with lower cost than the traditional separate approach. In fact, the integrated approach can tolerate as much as 15%- 60% lower resource availability than the separate approach. Moreover, most requests have at least the 98th percentile rank and can be found in 6 seconds with a population of 1 million hosts using standard SQL technology on today's hardware.

3.2 Slotted Virtual Grids: Incorporating Time into vgDL and vgES

We are currently developing an enhanced virtual grid architecture that allows users to specify, find, and bind resources across time. Our goals include adding support for space-shared (batch queues) and provisioned (lease-based) resources. We call this the slotted virtual grid (SVG), and it accommodates resources that arrive and depart according to information in the virtual grid request. This collection of enhancements supports dynamic workflow scheduling across resources available at different points in time.

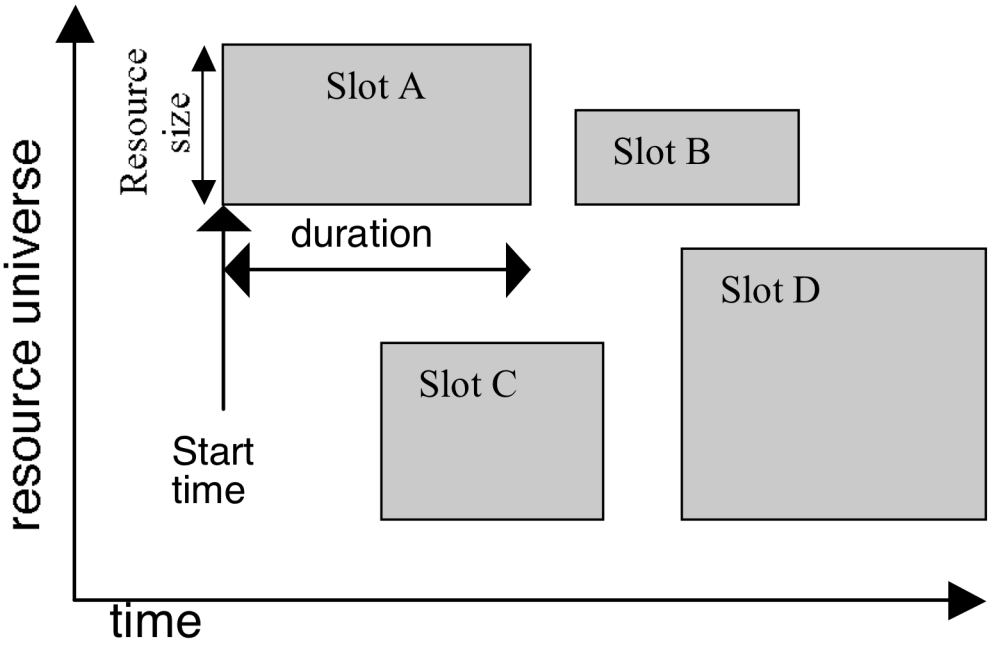


Figure 1 Four slots in a virtual grid request.

The virtual grid request describes how resources arrive and depart through time; it may specify a slot for any resource set. A slot is a window of contiguous time defined by a

{start, duration} tuple. Figure 1 illustrates a virtual grid request with four slots. The height of the slot represents resource quantity while its length represents how long the application requires those resources. The virtual grid execution system (vgES) provides resources that meet the constraints of kind, number, and slot. vgES delivers resources to the user when the slot begins and removes them when the slot ends. Resource arrival is equivalent to a bound resource; it is available for job submission or interactive use through an ssh-like facility.

vgES provides the underlying resource management so that resources arrive and depart according to the slot schedule. The underlying resources for a slot may come from one or more sites, hosting facilities or data centers. vgES interacts with site resource managers, a service provider that grants access to resources for some interval of time, to bind resources for a particular slot. For various reasons, resources may arrive early or late; vgES provides the necessary upcalls to inform the user of resource arrival and departure. Extensions to the virtual grid APIs allow the user to adjust their resource requirements in the face of these changes.

We are developing the following components to support the slotted virtual grid:

- Refine vgDL to allow users to specify resource slots.
- Add facilities to the underlying resource finding algorithms to incorporate slots as an additional search constraint. Update VG API to maintain a virtual grid across node arrivals, departures, and schedule disruptions.
- Incorporate a *slot manager* to maintain resources across multiple sites and slots.

Our ultimate goal in this work is to better understand how resources can be provisioned automatically for use by applications. We have already begun discussions with researchers in the tools project to see how slots can be used for scheduling. We expect to use slotted virtual grids to support resource selection and management for LEAD, a sophisticated workflow application that orchestrates data collection and simulation experiments for mesoscale weather prediction.

3.3 Resource Allocation and Scheduling for Slot Based VgES

The focus of this work is to allocate resources and schedule workflow applications on the allocated resources in order to optimize a given performance criteria such as maximum finish time, reliability etc. In the current Grid model of batch queuing systems being used as resource management systems, the resource allocation and the task scheduling are tightly coupled and the quality of service is best effort in nature. There are two important aspects in which our work differs from the past research done in workflow scheduling on Grids. First, by decoupling resource allocation from task scheduling and promoting the reuse of allocations for executing multiple tasks, we can reduce the uncertainty involved in the execution of the workflow and the resulting performance. Second, the quality of the workflow schedule depends not only on the

scheduling algorithm but also on the resources available. Given a deterministic scheduling algorithm, we try to optimize the workflow performance by making intelligent resource allocation decisions. These decisions include selecting which resources to allocate and deciding when and how long to allocate them. This set of allocated resources also defines a virtual Grid for the user.

In order to make the problem tractable and formulate it as a discrete optimization problem, we have developed a cost based model for resource allocation and scheduling of workflow applications on the Grid. The model defines the resources available to the user as a set of allocations, each identified by a processor count, start time, and duration among other things. A scheduling policy *S* can be used to schedule the workflow over the set of resource allocations. Both the set of allocations and the schedule can be built incrementally. There is a cost associated with the set of resource allocations known as the allocation cost and a cost associated with the resulting schedule known as the scheduling cost. The total cost is a function of the allocation cost and the scheduling cost. The optimization problem is to minimize the total cost with the constraint that a feasible schedule exists for the workflow on the allocated resources. By using different cost formulations we can simulate different allocation and scheduling strategies. Preliminary results have shown that given a finite set of allocations to choose from, we can significantly improve the allocation efficiency without much increase in the maximum finish time of the workflow by using certain cost formulations.

3.4 Monitoring the Virtual Grid with Mortar

The execution system enables applications to find and manage resources through a powerful, dynamic abstraction. A key challenge for these systems is to monitor the health and status of their applications once they are instantiated across the virtual grid. A fundamental building block of these systems is the ability to continually query application-defined data streams produced across nodes. For example, application controllers must coordinate data delivery and job execution, monitor for faults, and manage resource consumption across compute environments that are inherently failure prone. To this end we have developed Mortar, a lightweight distributed stream-processing engine that allows applications to manage, transform, and acquire data generated across the virtual grid.

Users specify “expectations” of system behavior, and these form the basis for higher-level software components. For example, a monitoring system may write an expectation for some threshold of failed processes or an event-driven application controller may write one to detect workflow stage completion. Mortar verifies expectations using in-network aggregating functions and scalable instance overlays, static trees of nodes that orchestrate the in-network computation. In particular, we have developed techniques to make these computations accurate and responsive even during node and link failures.

We have compared Mortar to DHT-based systems that also support in-network aggregates (SDIMS, [Yalagandula, et. al. Sigcomm 2004]). In several failure scenarios, Mortar's instance overlays can reduce error by 50%, with an acceptable increase in total network load. We continue to explore refinements that further improve accuracy while reducing network load.

3.4.1 Fault Tolerance in the Virtual Grid

Grid applications have diverse requirements for performance and reliability that are difficult to enforce, given variability across grid resources. Although there are tools and mechanisms to monitor performance and ensure reliability (e.g., via replication and over provisioning, checkpoint/restart and other schemes), few tools allow users to express reliability policies from the application's perspective, map these to resource capabilities and then coordinate and enforce strategies. Thus, we have designed extensions to the Virtual Grid API to allow users to clearly articulate reliability expectations in addition to performance, in qualitative terms when specifying resource requirements. Specifically, the virtual grid, as virtue of this work, will

- allow applications to describe *collective qualitative reliability or specific quantitative requirements* for resource selection,
- *adjust* fault tolerance levels and expectations *at run-time*,
- allow applications to *register a callback*, where application intervention might be required when certain fault tolerance constraints are violated.

We have defined a high-level qualitative reliability metric space that can be used by users to request resources. The qualitative levels are mapped to well-defined quantitative levels of reliability in the virtual grid to enable runtime monitoring and adaptation. We expect that over time the exact definition of the levels may vary or evolve.

We have defined a 5-point qualitative reliability scale of Excellent (90–100 %), Good (80–89%), Satisfactory (70–79%), Fair (60–69%), Poor (59–0%). This scale is then used to define reliability associators and operators in the virtual grid description language that specify the reliability of a node and the network. Thus, a user is able to specify that he or she would like a HighReliabilityBag of 16 nodes that connects with medium reliability to another LowReliabilityBag of 32 nodes.

We have also defined a simple constraint and policy language that allows applications to specify conditions under which a callback might be necessary or specify an action that needs to be applied under some set of constraints. An example policy might suggest that if the application is using more than 16 processors and the network reliability level

falls below “medium,” the task should be over-provisioned. We are experimenting with various policies that might work in this environment. The VG execution system will build on the integrated interface to Network Weather Service (NWS) and UNC's extant Health Application Programming Interface (HAPI) toolkit (developed previously) to collect fault-indicating data that will drive the fault tolerance strategy within the execution system.

3.5 Reasoning Framework for Performance Validation

We are also 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. The temporal reasoning framework supports grid environments to (a) monitor and validate the behavior of scientific workloads, (b) diagnose possible sources of behavior changes, and (c) provide reasoning support for applications to adapt to variations observed.

In the reporting period, we have focused on defining and prototyping the framework. The first step of the framework generates a qualitative behavior signature for the application. For each application execution, we capture the time series from the metrics of interest and generate a qualitative temporal signature. The collection of the qualitative temporal signatures captured from many executions of different scientific application constitutes the input to the *k-means* clustering algorithm. An online monitoring system captures the real-time data and assigns and uses class confidence to validate the behavior against the signature. The final component diagnoses the cause of unexpected behavior using behavioral interaction algebra and a temporal diagnosis reference space.

3.6 Simplifying Application Scheduling

Though grid applications now have access to increasing numbers of resources, they are finding it more difficult to take advantage of them. First, an increasingly large resource universe means that even polynomial time scheduling heuristics take a prohibitively long time to compute a schedule. Moreover, it may not be possible to gather all the resource information needed by a scheduling algorithm in a scalable manner. Our claim is that simple scheduling algorithms are sufficient to achieve good workflow performance, if they operate across an intelligently *pruned* resource universe. By leveraging the virtual grid abstraction (VG), we have developed a scalable two-level approach to scheduling workflows across federated system resources. The first phase whittles the resource universe down to a smaller resource subset that contains the most relevant resources for this workflow. The second phase uses a simple greedy scheduling heuristic.

Simulations across a range of typical DAG structures and resources demonstrate that this two-level approach is as effective and more scalable than existing complex heuristic DAG scheduling algorithms.

4 Project Milestones

As recommended by the NSF Site Review team (4/28-29/05), the VGrADS Principal Investigators have examined the original research milestones for the project. As is to be expected for a research project, some adjustments were needed based on initial experience and in changes in personnel (most notably, Andrew Chien's departure to become Chief Scientist for Intel). Below, we propose a revamped set of milestones.

4.1 Year 3 Milestones

The original milestones were

i. Execution System/Virtualization:
   - Novel resource selection and virtual scheduling strategy experiments with application kernels on virtual grid environments

ii. Execution System/Performance Provisioning:
   - Limited tunable performance/fault-tolerance capabilities

iii. Execution System/Grid Economy:
   - Begin designing experiments to test pricing techniques using VGrADS framework.
   - Continue simulation experiments to evaluate resource allocation efficiency.

iv. Execution System/Fault Tolerance:
   - Consider novel techniques

v. Programming Tools/Abstract Parallel Machine:
   - Prototype hybrid performance modeler for virtualized tasks

vi. Programming Tools/Abstract Component Machine:
   - Prototype library installer
   - Initial validation of library installer and mapping strategies with application

vii. Applications:
   - Produce a revised and usable instrumentation of one application (EMAN) sufficiently robust for independent use by application scientists

viii. Education, Outreach, and Training:
   - Continue AGEP program
   - Participate in Grace Hopper Conference

We have combined the virtualization and performance provisioning areas, reflecting the new directions that vgES is taking under Carl Kesselman's leadership. Similarly, we

have focused our tools efforts on workflow applications, reflecting our experience with EMAN and LEAD.

Our revamped milestones are

i. Execution System/Virtualization:
   - Investigate novel vgES resource selection and binding techniques and evaluate via simulation of large-scale environments
     (Done: vgES initial implementation and evaluation reported in HPDC-15 and CCGrid06 conferences, and Section 3.1 above)
   - Develop and evaluate techniques for scheduling in VG environments, exploiting synergy between resource specification and scheduling
     (Done: paper in IPDPS'06 and others under submission, discussed in Sections 3.6 and 2.1 above)
   - Augment vgES to include time-dependent descriptions via probabilistic and reservation-based provisioning
     (In progress: this is the genesis of Slotted Virtual Grids, reported in Sections 3.2 and 3.3 above)
   - Conduct experiments with the time-space reasoning contracts in virtual grid environments
     (In progress: discussed in Section 3.5 above)

ii. Execution System/Grid Economy:
   - Begin designing experiments to test pricing techniques using VGrADS framework.
     (Done: Batch-queue and reservation system pricing mechanisms under development.)
   - Enhance availability and batch-queue predictions to be suitable for scheduling, particularly with respect to virtual reservations.
     (Done: results reported in PPoPP'06 conference)
   - Continue simulation experiments to evaluate resource allocation efficiency.
     (Done: paper in HPDC-15 and others submitted)

iii. Execution System/Fault Tolerance:
   - Consider novel techniques and integrate predictive tools
     (Done: see Sections 1.4 and 2.5 above)
   - Develop a multi-level fault tolerance API to capture and adapt to failures at different levels and integrate with the virtual grid
     (Done: see Section 3.5 above)

iv. Programming Tools/Workflow:
   - Evaluate and refine multi-level scheduler(s) for workflow applications
     (Done: paper accepted at IPDPS'06 and other papers submitted, also discussed in Section 2.1 above)

- Adapt workflow scheduling to take advantage of new time-dependent descriptions available in vgES
  (In progress: to be developed in conjunction with LEAD integration)
- Explore novel workflow schedulers incorporating new resource behaviors (e.g. batch queue delays)
  (Done: formed one chapter of Mandal's thesis, with other papers under submission, also noted in Section 2.1)

v. Applications:
- Produce a revised implementation EMAN with queue-based scheduling enabled
  (Done: demonstrated at SC'05)
- Collaborate with LEAD application team on providing new capabilities for resource scheduling
  (Done: workshops with LEAD continuing)
- Evaluate fault-tolerance and virtual grid APIs with LEAD application
  (Done: workshops with LEAD and resulting integration architecture)

vi. Education, Outreach, and Training:
- Continue AGEP program
  (In progress: three AGEP students to be supported in summer 2006)
- Participate in Grace Hopper Conference
  (In progress: travel support and activities expected in August 2006)
- Continue program of grad student exchange and collaboration
  (In progress: series of implementation workshops in 2006)

4.2 Year 4 Milestones

Our original milestones were

i. Execution System/Virtualization:
- Improved Resource virtualization and virtual scheduling approaches based on experiments with additional virtual grid environments

ii. Execution System/Performance Provisioning:
- Extended tunable performance/fault-tolerance capabilities

iii. Execution System/Grid Economy:
- Convert GridSAT to use the pricing-based resource allocation.
- Conduct empirical investigation of pricing scheme and its effect on resource allocation stability using GridSAT as driving application.

iv. Execution System/Fault Tolerance:
- Implement novel techniques (e.g. diskless checkpointing)

v. Programming Tools/Abstract Parallel Machine:
- Initial application validation

vi. Programming Tools/Abstract Component Machine:
- Prototype library installer with component optimization

vii. Applications:

- Adapt the instrumentation strategy and its implementation to expected refinement in the VGrADS software environment. Initiate instrumentation of a second application and evaluation thereof

viii. Education, Outreach, and Training:
- Continue AGEP program
- Participate in Tapia Symposium

We continued the merging of Execution System and Programming Tools sub areas, and refined the outstanding bullets. We also added collaboration with the TIGRE grid, a project funded by the State of Texas to build a production grid infrastructure. (Rice and UH are partners in TIGRE.) This produced the following milestones:

i. Execution System/Virtualization:
- Evaluate resource selection, binding, and scheduling techniques based on time-dependent provisioning for VGs developed in Year 3
- Explore techniques to extend the VG abstraction to diverse resource management paradigms (e.g., probabilistic space-time resource abstractions).
- Expand techniques for integrated time-space reasoning with performance/fault tolerance capabilities

ii. Execution System/Grid Economy:
- Complete integration of new prediction capabilities with vgES to support both scheduling and grid economy work.
- Conduct empirical investigation of pricing scheme and its effect on resource allocations.

iii. Execution System/Fault Tolerance:
- Implement novel techniques (e.g. diskless checkpointing in a general setting in Open-MPI)
- Prototype and experiment with techniques to implement dynamic adaptation in a multi-level fault tolerance environment on the virtual grid

iv. Programming Tools/Workflow:
- Demonstrate novel, scalable workflow scheduler(s) on TIGRE grid
- Explore robustness of workflow schedulers, particularly with regard to fault tolerance

v. Applications:
- Incorporate novel, scalable workflow scheduler(s) into EMAN and LEAD applications
- Continue evaluation of fault tolerant techniques with LEAD

vi. Education, Outreach, and Training:
- Continue AGEP program
- Participate in Tapia Symposium
- Continue graduate student exchanges and collaborative workshops

4.3 Year 5 Milestones

Our original milestones were

i. Execution System/Virtualization:
   - Continue to improve Resource virtualization and virtual scheduling approaches based on more application kernel experiments
   - Integrate Resource Virtualization and Abstraction Classes into VGrADS software
   - Integrate virtual scheduling strategies into VGrADS software

ii. Execution System/Performance Provisioning:
   - Limited validation and assessment

iii. Execution System/Grid Economy:
   - Design experiment to investigate allocation efficiency under various pricing schemes.
   - Target second VGrADS-enabled application (to be determined) as a driving application.
   - Verify using both GridSAT and second application.

iv. Execution System/Fault Tolerance:
   - Limited validation and assessment

v. Programming Tools/Abstract Parallel Machine:
   - Prototype integration of fault-tolerance into mapper constructor
   - Studies of integration of grid economies into mapper construction.
   - Programming Tools/Abstract Component Machine:
   - Prototype domain-specific optimizations for library-based application

vi. Applications:
   - Produce an instrumentation of both the first and second application usable by application scientist for the VGrADS software in its state the final year. Document instrumentation strategies and tools in a way suitable for application scientists.

vii. Education, Outreach, and Training:
   - Continue AGEP program
   - Participate in Grace Hopper Conference

Continuing with the refinements started in year 3, our updated milestones are

i. Execution System/Virtualization:
   - Improve resource selection and binding techniques for VGs based on insights from Year 4 evaluation
   - Improve resource scheduling techniques for VGs based on insights from the Year 4 evaluation
   - Prototype and evaluate extended VG abstraction over environments with diverse resource management paradigms.

- Demonstrate vgES with resource selection, resource binding, VG scheduling, for application kernels across large-scale grid platforms with diverse resource management paradigms.
- Validate and assess the integrated resource provisioning policies

ii. Execution System/Grid Economy:
- Design experiment to investigate allocation efficiency under various pricing schemes.
- Target second VGrADS-enabled application (to be determined) as a driving application.
- Verify using both GridSAT and second application.

iii. Execution System/Fault Tolerance:
- Limited validation and assessment

iv. Programming Tools/Workflow:
- Incorporate novel techniques from vgES and fault tolerance work into workflow schedulers
- Study new problems in workflow suggested by experience in years 3 and 4.

v. Applications:
- Evaluate EMAN and LEAD capabilities on large-scale TIGRE grid
- Extend application research as suggested by year 3 and 4 experience

vi. Education, Outreach, and Training:
- Continue AGEP program
- Participate in Grace Hopper Conference
- Continue graduate student exchanges

II. Findings

During the reporting period (6/1/05–5/31/06), VGrADS research focused on three inter-institutional efforts: *Applications, VGrADS Programming Tools,* and *VGrADS Execution System*. The following sections summarize the findings of each subproject.

5 Applications (Rice, UCSD, UCSB, UH, UTK, UNC)

Work in this reporting period produced the following findings for the four applications.

EMAN: EMAN showed that scheduling workflow applications for efficient execution on real Grids consisting of non-dedicated machines not only requires relatively accurate estimates of node performance but also accurate predictions of queue-wait times on the Grid resources. Accurate estimates of both result in better turnaround time for the EMAN computations.

LEAD: LEAD, in addition to the traditional resource requirements for running an ensemble of simulations – tightly connected clusters and loosely coordinated larger collections – also requires streaming data. LEAD has different use cases, including research and educational interaction with the portal and real-time launch of computational elements. These different interactions affect the resource scheduling policies, which must be studied further. LEAD also adds an explicit requirement for fault tolerance, which necessitates options for trading fault tolerance for output fidelity.

FT-LA: Two prototype examples have been developed to demonstrate the effectiveness of our techniques. In the first example, we developed a fault survivable conjugate gradient solver that is able to survive multiple simultaneous process failures with negligible overhead. In the second example, we incorporated our checkpoint-free fault tolerance technique into the ScaLAPACK/PBLAS matrix-matrix multiplication code to evaluate the overhead, survivability, and scalability. Theoretical analysis indicates that, to survive a fixed number of process failures, the fault tolerance overhead (without recovery) for matrix-matrix multiplication decreases to zero as the total number of processes (assuming a fixed amount of data per process) increases to infinity. Experimental results demonstrate that the checkpoint-free fault tolerance technique introduces surprisingly low overhead even when the total number of processes used in the application is small.

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.

Schedulers: Although our experiments continue, it appears that Virtual Grids provide important benefits to application scheduling. Selection of resources through vgDL reduces the complexity of the schedule, in both the theoretical ($O(v^2p)$, where p is the number of processors) sense and the ease-of-use sense. Schedules computed from reduced VGs are competitive with schedules computed on the full Grid universe, and in some cases better.

We have shown that precomputed schedules are practical, even in a dynamic context like the Grid. The benefits of intelligent scheduling often outweigh the inaccuracies due to inaccurate performance models. Moreover, we have demonstrated accurate performance models suitable for the Grid.

Our preliminary results also indicate that batch queue wait time predictions are feasible, and can be incorporated into workflow scheduling algorithms. This allows precomputed schedules to take advantage of resources controlled by batch queues, a significant extension to previous practice.

Fault Tolerant Libraries: We have shown that high-performance fault-tolerant libraries are possible in a Grid context. This work is contributing to high-quality open-source software to improve the infrastructure for distributed system applications.

6 VGrADS Execution System (UCSD, UCSB, UTK, UNC, ISI)

vgES Impact: We have released the software to other internal team members; those teams are developing advanced workflow scheduling techniques on top of the VG and vgDL abstractions. Indeed we have found both the vgDL language and the vgES system to be useful abstractions for two-level resource scheduling, providing a clear benefit over traditional methods, which attempt to schedule against an entire universe of resources, as opposed to a paired-down search space that retains the most pertinent resources.

Our 2-level scheduling results, published in IPDPS'06, show that simple scheduling algorithms may be sufficient to achieve good workflow performances. The impact of our finding is clear for scheduling grid workflows in practice: rather than investing time in developing and implementing sophisticated scheduling algorithms, one should initially implement simplistic algorithms but perform fast and appropriate resource pre-selection.

Integrated Finding and Binding: The key finding is that resource selection should be lightweight and aware of binding. Results reported in a research paper submitted to SC'06 show that the integrated selection and binding can allocate resources of significantly higher quality at higher success rate and lower cost than the traditional separate selection and binding approaches in shared resource environments.

Federated System Monitoring: We have found that the expectation abstraction is a powerful basic system building block. We built an application controller for a grid-enabled scientific application, a basic monitoring system, and an adaptive scheduler using 5 expectations. Additionally, our large-scale experiments validate the robustness and accuracy of our techniques for implementing in-network functions. The expectation abstraction and experimental results are in submission to SC'06.

III. VGrADS Education, Outreach, and Training Activities

The following sections describe VGrADS Education, Outreach, and Training (EOT) activities during the current reporting period (6/1/05-5/31/06).

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. During the summer of 2005, Ryan Zhang, a first-year student at Rice University, spent two months at the University of California San Diego (UCSD). He engaged in research with VGrADS personnel at UCSD, primarily aimed at more closely linking the programming tools group with the execution system group. Among the outcomes were improvements (bug fixes) to vgES to enable a variety of scheduling experiments and new ideas in scheduling workflow applications. Also in 2005, Dan Nurmi, a UCSB student, spent three weeks at Rice University collaborating on scheduling and performance estimation for batch-queue-controlled systems. His efforts led directly to the new batch queue scheduler described in the research contributions section.

In 2006, our plans for student exchanges have shifted somewhat. Because we have embarked on a major integration project – running the LEAD application under

VGrADS – we felt that better communication between all groups was needed. We have therefore scheduled a number of smaller workshops for students (and some staff) involved in technical collaboration. These are in addition to our "usual" semi-annual workshops for high-level discussions and planning. The small workshops are truly working meetings, producing detailed plans and software artifacts for use in our experiments and demonstrations. The first working meeting was held at UNC in April, and resulted in the first descriptions of slotted virtual grids (SVGs, see Section 3.2 of *Project Activities*), schedulers for SVG systems, and a detailed architecture and work plan for producing the VGrADS/LEAD integrated system by SC'06. The students involved, of course, received significant experience in collaborative work, distributed software development, and a broader exposure to the project than they would ordinarily have had.

1.3 Distributed Software Engineering

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, VGrADS PIs have developed and taught a variety of courses that cover Grid technologies and other aspects of high performance parallel and distributed computing. For example, in the Spring 2006 semester, Jack Dongarra taught *CS 594-001: Understanding Parallel Computing* at UTK. For more information on this course and its contents, visit http://www.cs.utk.edu/%7Edongarra/WEB-PAGES/cs594-2006.htm.

2 Outreach Activities

The Outreach component of VGrADS has continued its efforts to broaden the impact of the project.

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 that funds a number of activities at Rice (and other universities) to provide a year-round community experience for Science/Math/Engineering (SME) students from

under-represented groups. Most relevant for VGrADS activities is the Rice AGEP summer program, which 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 serving as mentors of these summer students. AGEP leverages VGrADS to reach more students, through our direct funding of additional participants.

In summer 2005, we sponsored two female African-American students to research Grid-related problems. (We accepted a third student, but she took a position elsewhere.) Both benefited professionally from the experience, particularly from the interactions with their peers and with Richard Tapia. Unfortunately, the experience was not as positive from the research perspective. The students' academic background (from non-research universities) did not prepare them well to write even simple programs. Coupled with loose oversight from their mentor, this produced no tangible results for the research effort. We have learned from this experience, and have refined our recruiting strategy accordingly.

Recruiting for the summer of 2006 is nearly complete. Approximately 100 students have applied to the AGEP program, of whom more than a dozen have specifically expressed interest in working with VGrADS. We have accepted three students, all female (one Hispanic). Two of the students are from Carnegie-Mellon, a top-tier research university, while the third is from University of Houston Downtown, a local minority serving institution. All have accepted the positions. We have provided all of them with Grid-related books to jump-start the research process, and we look forward to an interesting summer.

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. Both meetings are biannual, with the Tapia Conference meeting in odd-numbered years and the Hopper Conference in even-numbered years.

In 2005, VGrADS provided travel support to three staffers from VGrADS sites - Chuck Koelbel from Rice, Shava Smallen from SDSC/UCSD, and Diane Pozefsky from UNC - to the Tapia Celebration, held in October in Albuquerque, NM. Fran Berman, a VGrADS PI, was a plenary speaker at Tapia, talking about distributed scientific

applications (although not specifically about VGrADS). Koelbel, another VGrADS PI, chaired a panel on grid computing that featured Smallen and Pozefsky.

Usually it is hard to show concrete outcomes from supporting outreach-based conference participation. This year, though, we can point to a real success from past VGrADS activities. In 2004, VGrADS supported Amanda Cruess, a Rice undergrad, to attend the Grace Hopper conference. In May 2006 Amanda received an Impact Award from Rice's Women's Resource Council for supporting and promoting the contributions of women in computing. This was not merely a happy coincidence. As Amanda herself said when receiving the award, "It was the Grace Hopper conference that really inspired me to get more involved with CSters [Rice University's women in computer science group]. It was incredible to see all of these women there, all doing exciting things in the field of computer science." For more information about the award and its recipient, see http://cohesion.rice.edu/engineering/computerscience/news.cfm?doc_id=8854.

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

As we have since our inception, VGrADS PIs participated in the NSF-funded Computer Science Computer and Mentoring Partnership (CS-CAMP) project (http://ceee.rice.edu/cs-camp/). Four VGrADS PIs (Richard Tapia, Keith Cooper, Ken Kennedy, and Chuck Koelbel) made presentations at CS-CAMP sessions in 2005 for high-school CS teachers and high-school girls. Ken Kennedy's talk, *Parallel Computing and Grids*, was specific to grid computing. The other presenters spoke on a variety of other computing and diversity topics, including careers and research in CS. VGrADS PIs Keith Cooper and Richard Tapia are PIs of the CS-CAMP project.

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 SC'05 Conference (Seattle, WA, November 12-18, 2005). 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) gave many overviews of the VGrADS project before a variety of demonstrations.
- Anirban Mandal (Rice), Dan Nurmi (UCSB), and Rich Wolski (UCSB) demonstrated batch queue scheduling.

- Jack Dongarra and Asim YarKhan (both UTK) demonstrated the GridSolve system.
- Andrew Chien, Yang-Suk Kee, and Jerry Chou (all UCSD) demonstrated the latest features of vgES (execution system) and vgMON (grid monitoring).

Most of these demos attracted reasonable audiences, and represented good outreach to the high-performance computing and research communities by the project.

2.5 Other PI Involvement in Education and Outreach Activities

VGrADS PIs continue to be involved in a variety of education and outreach efforts that are not directly related to VGrADS. Since the inception of VGrADS, PIs have been involved in efforts to increase diversity within their institutions. They have also participated in a variety of high-profile education and outreach efforts. Examples of activities during the funding period include:

- Keith Cooper (Rice) is 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. One two-year fellowship was awarded in Fall 2005; another two-year fellowship will be awarded in Fall 2006.
- Dan Reed (UNC) presented "*Computing: A 21st Century Liberal Arts Education*" at the CRA Workshop on the Future of Computing Education in November 2005. The talk emphasized the need to nurture and energize the next generation of computer scientists as liberal arts specialists that will work on inter-disciplinary problems to solve the toughest scientific, economic and social problems.
- Dan Reed (UNC) talked about "*The Networked Learning Environment*" at the North Carolina Distance Education Conference. He talked about the role of the university in the current information and illustrated some successful distance education programs that utilize new technologies in their curriculum.
- Dan Reed (UNC) has been appointed to the President's Council of Advisors on Science and Technology, where he co-chairs the subcommittee on information technology.

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 continued to develop 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, our (admittedly preliminary) experiments indicate that a two-phase strategy of a simple virtual grid selection phase (picking "good" resources to run on) followed by relatively simple scheduling heuristics (such as greedy scheduling) obtains very good overall application performance. Using more sophisticated schedulers (such as list scheduling) in the second phase is also promising.

- VGrADS researchers have also extended the range of applicability of a priori schedulers for computational Grids. We have shown that it is possible to combine estimations of application performance and batch queue wait times to generate high-quality schedules. We are adapting this work to schedule major computational phases for the LEAD application. We have also developed schedulers that simultaneously optimize time to completion (makespan) and robustness of the resulting schedule.

- VGrADS researchers have studied the feasibility of traditional compiler optimizations such as register allocation in the context of just-in-time compilation. This led to new register allocation heuristics that met the strict performance needs of a JIT setting, yet still produced significant improvements in execution time. This work supports the VGrADS philosophy of using local compilation to enable heterogeneous executables.

- VGrADS researchers have developed the FT-MPI library to provide process-level fault tolerance based on the MPI 1.2 standard. The performance of the resulting implementation is comparable to MPICH2 or LAM, even with the fault tolerance features enabled. This work is being incorporated in the OpenMPI project, which is creating a completely new MPI-2 implementation using the best library technologies and resources available.

- VGrADS researchers have continued development of the Virtual Grid Execution System (vgES) for managing the abstractions that are key to our work. A particular contribution this reporting period has been the development of the Mortar monitoring system, which uses "expectations" to efficiently detect and report the health of the VG. In particular, we have developed techniques to make these computations accurate and responsive even during node and link failures.

- VGrADS researchers are building on the integration of HAPI and the NWS system to provide qualitative and quantitative information about the reliability of Virtual Grids. We are experimenting with policies that use this information to better manage application fault tolerance.

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, VGrADS PIs continue to develop and teach a variety of courses that cover Grid technologies and other aspects of high performance parallel and distributed computing. For example, in the Spring 2006 semester, Jack Dongarra taught *CS 594-001: Understanding Parallel Computing* at UTK.

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 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 SC2005 conference in Seattle, WA (November 12-18, 2005). VGrADS activities at SC2005 are discussed under both "Outreach Activities" and "Project Activities."