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. The Virtual Grid Execution System (vgES) implements this architecture.
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 is also building on its PIs' past successes in human resource development by leveraging existing programs to attract and retain women and minorities in computational science.

During the current reporting period (6/1/07 – 5/31/08), 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. The management structure of the VGrADS project is described in *Management & Structure (Section 4)*. Annotated VGrADS milestones appear in *Project Milestones (Section 5)*.

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

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. We have subsequently completed our study of EOL. Some research in the area of fault tolerance has been done on another application called GridSolve. To date, these applications have been reasonably successful at helping to derive requirements for the VG functionality and vgES implementation. We summarize recent work on each application in the following four subsections.

1.1 LEAD

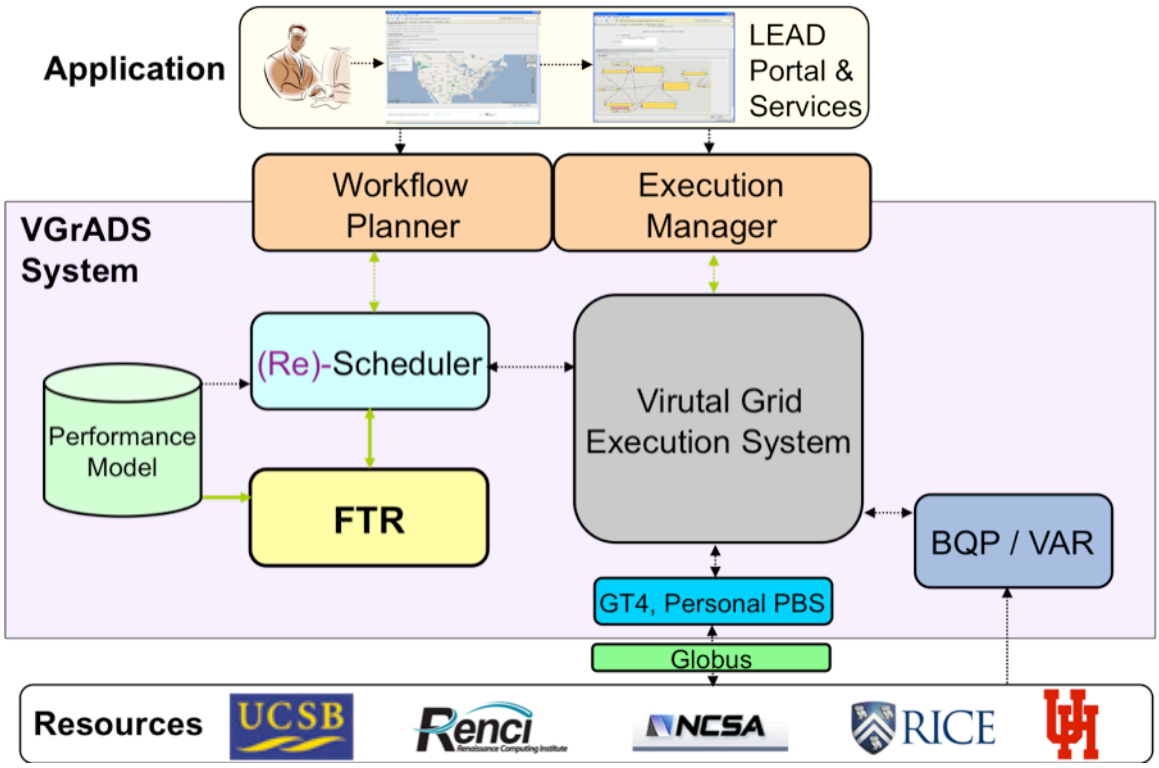
VGrADS is closely collaborating with another NSF funded ITR project: LEAD (Linked Environments for Atmospheric Discovery). The goal of LEAD is to build a scalable web services workflow infrastructure for meteorological data and models. The meteorology and the web services components of LEAD are being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

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 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 have collaborated with the LEAD developers to develop integrated plans for the two projects. This process has been greatly expedited by the efforts of Lavanya Ramakrishnan, a former VGrADS programmer at RENCI who is now a PhD student with LEAD PI Dennis Gannon, and Anirban Mandal, who is currently at RENCI. They have organized a number of meetings and teleconferences, and generally led the development of the LEAD/VGrADS integration.

As noted in previous annual reports, we have had meetings with LEAD personnel to map out a collaborative strategy. A number of technical issues had to be resolved, mostly relating to how VGrADS would interoperate with LEAD's existing production software. Also, we had to address the fact that LEAD's definition of workflow is subtly different from the definition used by other VGrADS applications. 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 web service, whereas tasks in other VGrADS applications are not. In addition, there is a need for managing streaming data, as opposed to the fixed data collections used in the other applications.

In this reporting period, we successfully incorporated fault-tolerance and runtime adaptation functionalities into the VGrADS/LEAD integration software stack to enable resilient execution of LEAD workflows. Resilient workflow execution is of paramount importance to LEAD workflows because of deadline constraints (forecasts have to be available by a certain deadline). We demonstrated these resilience features at SC'07. This work was built on top of our prototype VGrADS/LEAD integration that we demonstrated at SC'06. This work was a true collaboration, involving Lavanya Ramakrishnan (originally UNC, later moving to Indiana), Anirban Mandal (UNC), Gopi Kandaswamy (UNC), Yang-Suk Kee (ISI), Dan Nurmi (UCSB), Ryan Zhang (Rice), and many others. In this exercise we refined the resource slot concept, showed how the Virtual Advanced Reservation (VAR) mechanism could be used to probabilistically reserve and manage slots, demonstrated slot-based application scheduling based on performance models, and showed how different fault-tolerance techniques like over-provisioning and migration can improve the reliability of LEAD workflow executions. The net effect was that LEAD could produce an abstract workflow (as a DAG with constraints) and have the workflow scheduled and reliably executed on VGrADS and TeraGrid resources. The figure below shows the relevant components of the VGrADS/LEAD integration stack and the invocation flow for various components.



The following steps are executed to generate a fault-tolerant schedule:

1. The Workflow Planner and the Execution Manager serve as the interface between the LEAD and VGrADS systems. The Workflow Planner obtains the DAG and constraints (e.g. deadlines) from the LEAD portal and services.
2. The Workflow Planner passes the DAG, constraints, and a pointer to the application performance model to the VGrADS scheduler.
3. The VGrADS Scheduler queries the performance model for the task's resource requirements to meet the constraints.
4. The Scheduler requests slots from the Virtual Grid Execution System (vgES) using the `vgFind` command.
5. vgES queries the Virtual Advanced Reservation (VAR) system for the probabilities of getting slots. For reserved resources, the probability is 1 (ignoring node failures); for other resources, it is a statistical prediction.
6. Based on VAR data, vgES chooses slots and returns a representation to the scheduler.
7. The Scheduler assigns DAG nodes to slots, using the performance model to predict computation and communication time.
8. The Scheduler consults with the Fault Tolerance and Recovery (FTR) component, which determines the degree and resources for over-provisioning.

9. FTR uses performance models, availability probabilities of resources (via an availability prediction service), resource and network models and the DAG constraints to determine the best fault tolerance mechanism that induces reliable workflow execution.
10. FTR returns the degree and resources for over-provisioning a workflow step to the Scheduler.
11. The Scheduler iterates with FTR to determine a fault-tolerant plan for the entire workflow and returns the plan (mapping of (replicated) tasks to slots) to the Workflow Planner.
12. The Workflow Planner issues a `vgBind` command to vgES.
13. For batch-queued resources, vgES queries VAR (repeatedly) for time to submit job.
14. vgES acquires resources at partner sites. For reserved resources, Globus GRAM is used to glide in a personalized PBS server; for batch-queued resources, a job is submitted to the local PBS queue. This process must be followed for each slot to be bound.
15. Each slot is given a PBS queue, accessible via a Globus gateway.
16. When all slots are bound, the Workflow Planner annotates the DAG with slots and the LEAD workflow is ready for execution.

The following steps are executed for managing execution of the workflow:

17. The Execution Manager sends a `vgLaunch` command for the current workflow task to vgES.
18. vgES runs the task on the allocated resource gateway.
19. While the job executes, the Execution Manager issues `vgStatus` commands to monitor progress.
20. If the Execution Manager detects a failure for a workflow step, a re-scheduling is triggered.
21. The Execution Manager consults the Scheduler and FTR to generate a new fault-tolerant plan for the remaining portion of the workflow.
22. The new plan is returned to the Execution Manager and the unfinished portion of the workflow begins execution. The failed workflow step may be migrated to another resource in the process.
23. This continues until the entire workflow finishes execution and the data products are ready.

Here are some snapshots of a LEAD workflow execution that incorporates the fault-tolerance features. These snapshots show a LEAD workflow and how it was scheduled in a fault-tolerant manner over two TeraGrid resources and two VGrADS resources. They also show how different workflow steps have been over-provisioned across multiple resources to guarantee a higher probability of meeting the workflow deadline.

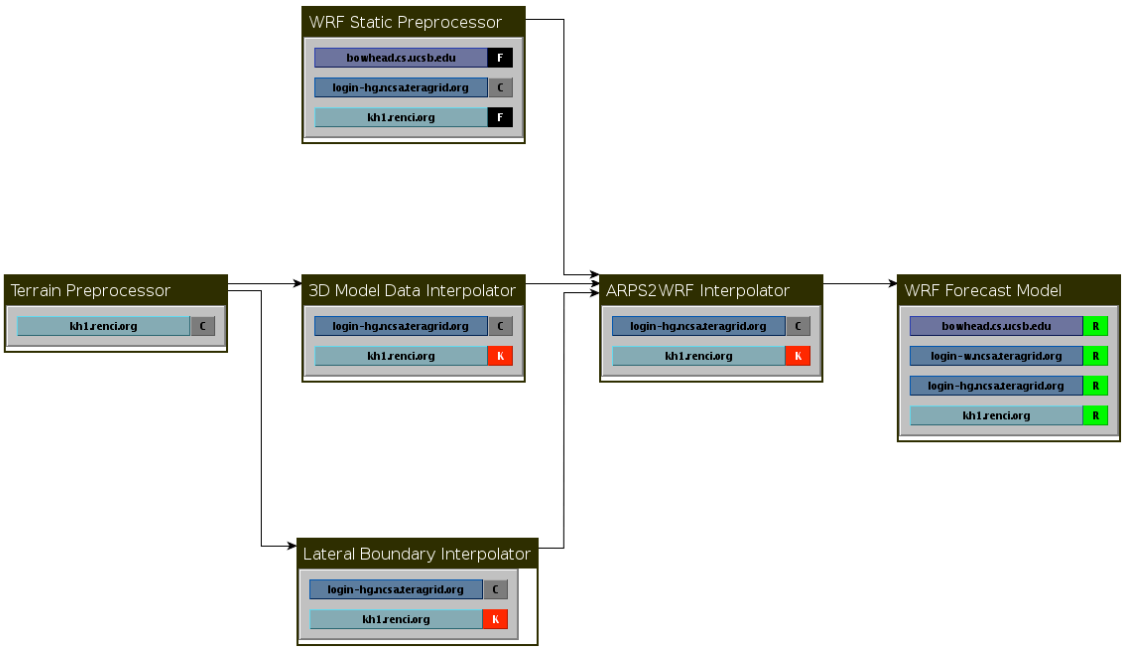


Figure: LEAD workflow and status of workflow steps

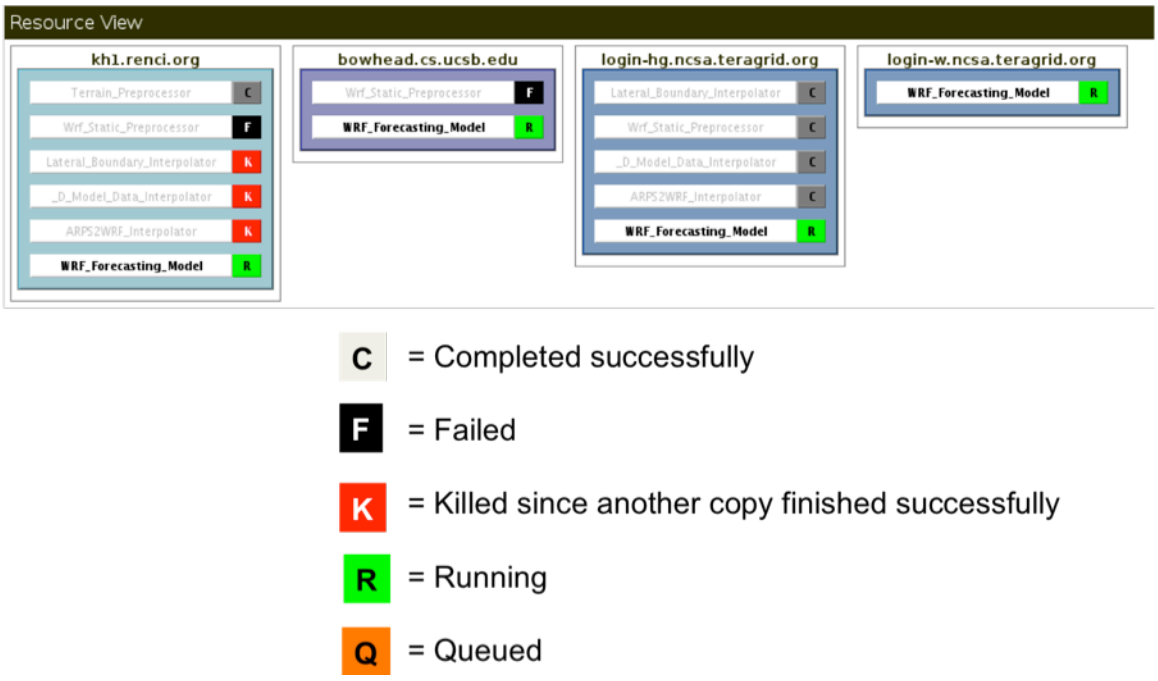
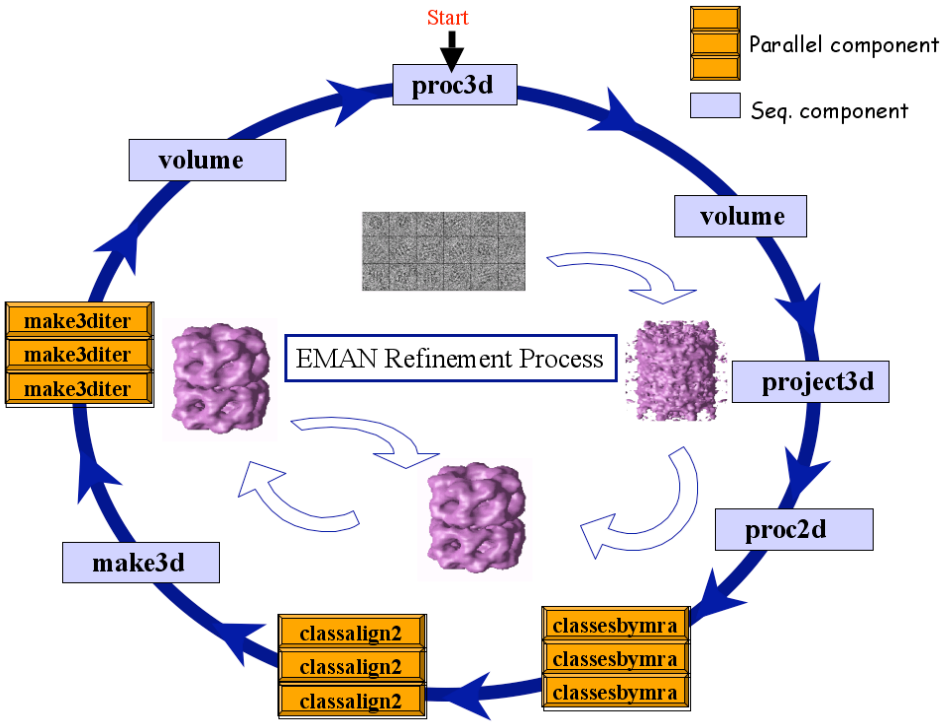


Figure: State of LEAD workflow steps across TeraGrid and VGrADS resources

1.2 EMAN

Much of the early VGrADS application research effort focused on the EMAN software (http://ncmi.bcm.tmc.edu/ncmi/homes/stevel/EMAN/doc), which we used 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.



The lessons of EMAN for virtual grids, scheduling, and workflow management have been incorporated into our LEAD developments. To summarize past work in those areas:

- EMAN's parallel phases are embarrassingly parallel computations (although they could, in principle, be reformulated as more tightly-coupled computations), coded in some cases to require a shared file system and in some cases with no coupling requirement. These motivated the original "TightBag" and "LooseBag" features of vgDL.

- EMAN parallel phases (without shared file systems) lent themselves to distribution between multiple clusters. This fact led to both the hierarchical vgDL structure ("LooseBag of Cluster") and two-phase scheduling. It was also the primary motivation for our development of Batch Queue Scheduling.
- EMAN workflow implementations require schedulers to be aware of both computation and communication, which has been a cornerstone of our work.
- EMAN was also the genesis of our early emphasis (as far back as GrADS) on performance models. The fact that certain phases of EMAN were extremely sensitive to the processor architecture (a factor of 3 difference, if clock speeds were the same) motivated much of our work in this area.

EMAN was also used as the test case for Batch Queue Prediction scheduling. The crux of this work was that our older scheduling methods assumed that all resources were dedicated to VGrADS work, which was not true even for our own experimental machines. Our solutions to these issues formed the basis of demonstrations at SC05 and thereafter, and are a large part of Ryan Zhang's thesis.

It also bears mentioning that EMAN workflow DAGs continue to be useful in our experiments, whether simulations or real-world runs. For example, EMAN was one of two actual application DAGs used in Ryan Zhang et al.'s CCGrid '07 paper and many of the conclusions therein were based on it.

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) that 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 a 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 can, in principle, handle this task more scalably. However, as GridSAT is already highly tuned for grid environments, incorporating VGs would be a strong test of vgES efficiency.

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 elsewhere.

GridSAT has fallen out-of-date as the student developer graduated.

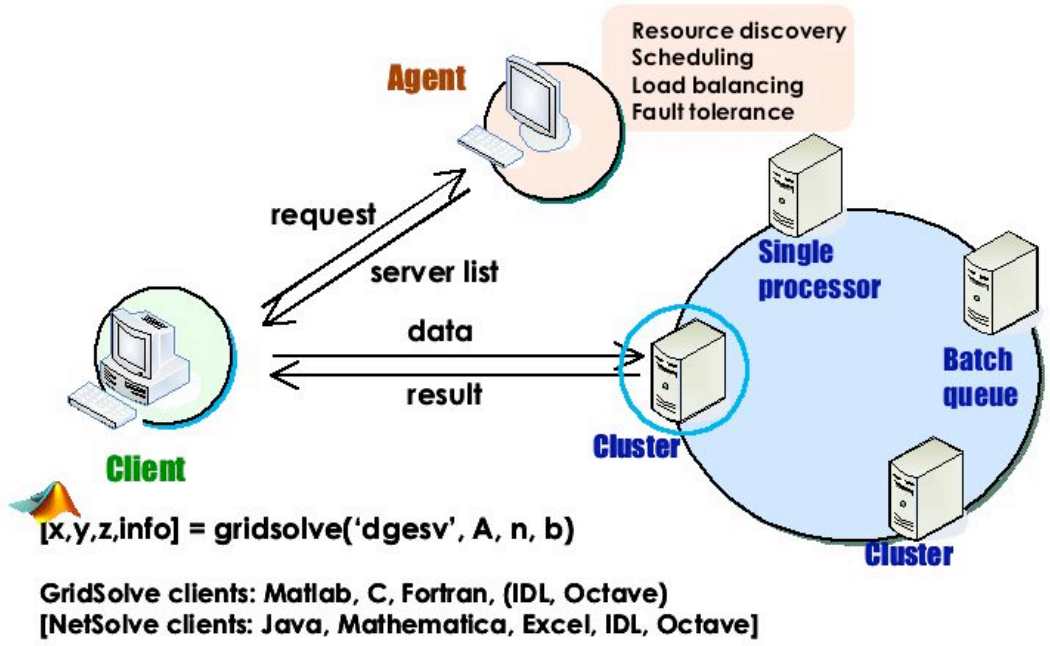
1.4 FT-LA: Fault Tolerant Linear Algebra

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; and (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.

1.5 GridSolve

UTK has a long-term interest in providing mathematical solvers to enable computational science. The latest of these software systems is GridSolve, an attempt to provide (through middleware) a seamless bridge between the simple, standard programming interfaces and desktop systems that dominate the work of computational scientists and the rich supply of services supported by the Grid. This allows computational scientists running Matlab on their desktops to reap the benefits (shared processing, storage, software, data resources, etc.) of supercomputers. GridSolve's design and implementation use a layered approach to manage interactions between the Grid's diverse management software (e.g. local schedulers) and the user community's equally diverse application structures (e.g. parameter sweeps, workflows, etc.). This design relies on a GridSolve agent to translate the user's high-level request into specific instructions for the grid resources, as shown in the figure below.



This design integrates well with virtual grids since, from the agent's point of view, vgES can be considered just another resource manager. However, using vgES rather than lower-level managers (e.g. Condor, PBS) allows the agent to collect resource information more accurately, and allows the agent to leverage VGrADS research. The investigations of GridSolve scheduling have been influential in our more general work on vgES, particularly in the area of managing fault tolerance.

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

The broad vision of the programming tools thrust is to provide for application users high-level interfaces that allow automatic construction of capabilities that are (currently) hard to achieve in a Grid environment. At the core of this work is our attempt to take advantage of the virtual grid (VG) abstraction and tools to provide more application-specific abstractions.

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

1. Improved scheduling for workflow computations on VGs in a variety of situations,
2. Developing economic methods for resource allocation,
3. Compiling and optimizing node programs for use in a Grid environment, and
4. Fault-tolerant libraries for MPI.

The following subsections discuss each thrust in turn.

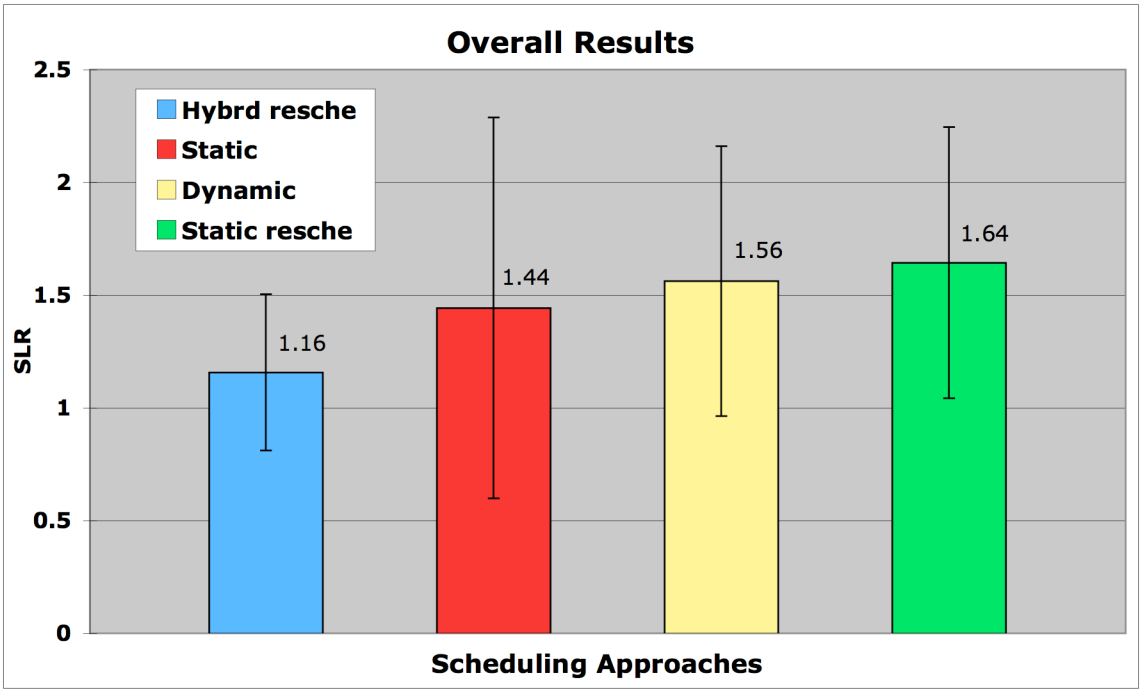
2.1 Scheduling Workflows on Virtual Grids

As we have previously reported, researchers at Rice, UCSD, and UNC have developed scheduling strategies for EMAN, EOL, and other workflow applications. These applications are modeled by directed acyclic graphs (DAGs), where the nodes are computations (which may themselves be parallel structures) and the edges are data communication. A key part of this has been the study of the "two-phase" scheduling strategy, where the first phase is selecting a VG for an application, and the second phase is mapping the application onto the chosen VG. This has been so successful that we have adopted it in nearly all of our ongoing work.

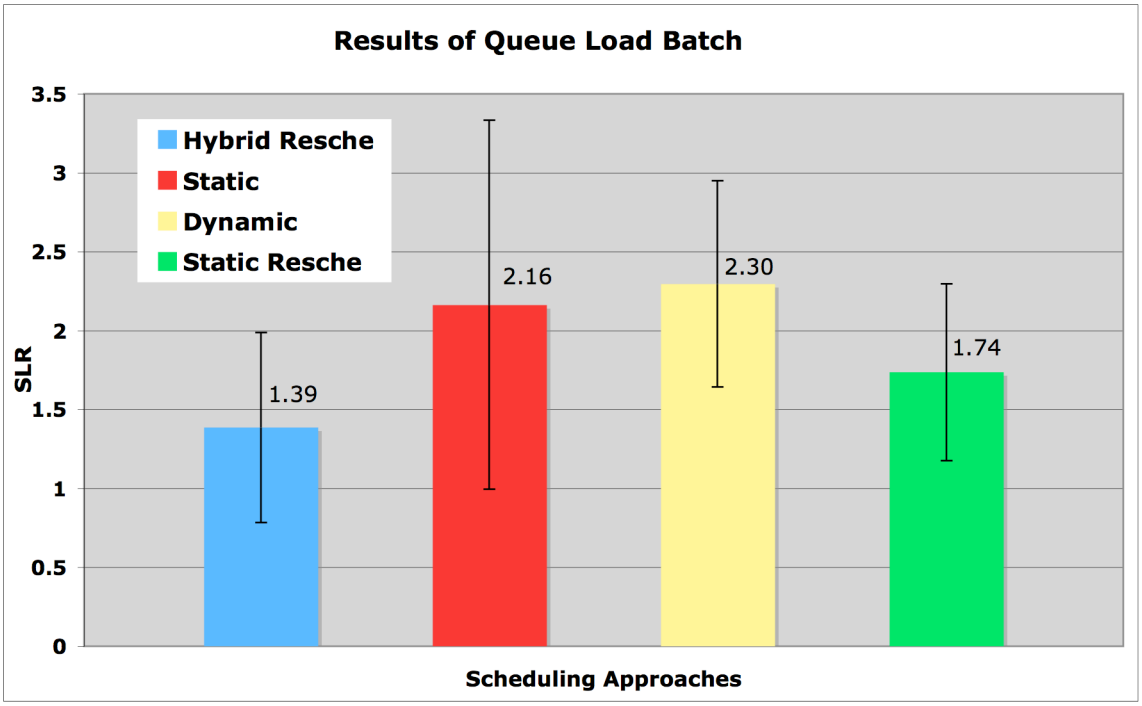
Last year we reported that our paper on this topic "Performance of Scheduling Algorithms in Grid Environments" by Ryan Zhang and VGrADS PIs Charles Koelbel and Ken Kennedy (all from Rice University) had been nominated as best paper at the CCGrid'07 conference. In the end, the conference committee presented a special "Best Sustained Technical Contribution Award to Prof. Ken Kennedy and his team at Rice University." This honored not only the 2007 paper, but also an entire series of papers (at least one accepted in every year of CCGrid since its inception) published by the VGrADS and GrADS projects. Scheduling methods studied in the 2007 paper were used in the SC07 demonstrations detailed elsewhere in this report.

Zhang and Koelbel have continued their work on workflow schedulers, having submitted a new paper to the Grid2008 Conference. Although this work has not yet been accepted for publication, we see it as an important step forward in VGrADS. A serious limitation of our previous work—and a source of concern for some reviewers—has been the static nature of the schedules versus the dynamic nature of Grid resources. The new work addresses this by developing a class of hybrid static/dynamic schedulers

that combine performance-model based *a priori* scheduling (as in our prior work) with dynamic performance monitoring and rescheduling. The chart below, taken from the submission, shows the improvement of the new method ("Hybrid resche") over our previous purely static method ("Static"), a greedy dynamic scheduler ("Dynamic"), and a static schedule with periodic rescheduling ("Static resche"). The methods are compared by the Schedule Length Ratio (SLR), a comparison to the estimated critical path length of the computation; higher SLR is better. The smaller bars show the standard deviation of the SLR. All computations were run multiple times on a real Grid.



Perhaps even more telling is the following chart from the same paper. This shows a subset of the runs in which we introduced an artificial load on the Grid by submitting extra jobs to the same batch-queued resources. Unsurprisingly, this produces very poor performance for the purely static method. However, it also affects the dynamic scheduling dramatically, perhaps because our dynamic method does not adequately consider queue wait times. In any case, these tests demonstrate that a hybrid method based on continuously updated performance models is worthy of continued study.



In addition to the new scheduling method, Zhang also presented his thesis proposal on scheduling Grid computations. New work in that proposal promises to unify the schedulers we have developed for batch-queued resources and scheduling for fault tolerance. If all goes well, Zhang should complete his thesis in the 2008-2009 school year.

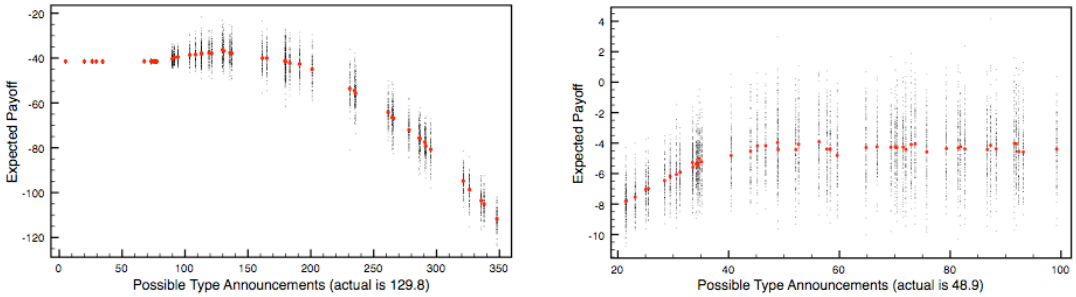
2.2 Resource Allocation via Grid Economies

As reported in previous annual reports, researchers at UCSB have embarked on a study of economics-based methods for resource allocation and scheduling on grids. The inspiration for this study was Ken Kennedy's observation that application performance models (also developed under GrADS and VGrADS) provide a measure of "value" of a resource to an application. He suggested that we use this as input to economic models that Rich Wolski and collaborators were developing.

Our current approach to this idea is the development of auction mechanisms for control of batch queued resources. A key problem in this space is extracting accurate information about the performance and importance of the individual jobs. Because the execution of one job prevents all others in the queue from using it, the economic "externalities" of scheduling a resource give users a great incentive to lie in order to get to the front of the queue. Even without evil intent, fallible humans often provide inaccurate information, either due to the difficulty of estimating performance or because

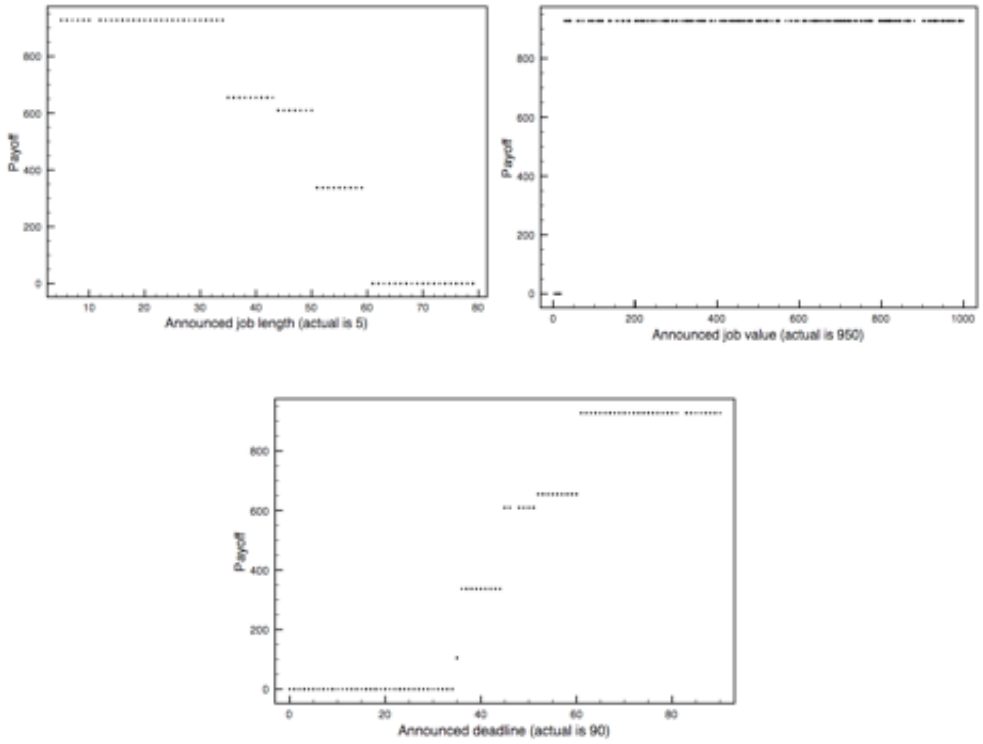
they do not fully understand the pricing mechanism. Such inaccuracies, of course, destroy any optimality properties of the scheduler.

Andrew Mutz, a student working under VGrADS PI Rich Wolski, has been investigating this problem. Their paper "Eliciting Honest Value Information in a Batch-Queue Environment", presented in the 8th Grid Computing Conference, describes a new scheduler based on the Expected Externality auction mechanism for scheduling jobs on batch-queued resources. In this scheduler, each job specifies its importance (v, in "currency" units), its execution time (t) and its wait tolerance (d, in "currency" per time). Whenever the system can accept a job, the head of the queue is examined. If its value v_i is greater than or equal to the tolerances of the remaining jobs in the queue ($\sum_{j \neq i} t_i \cdot d_j$), then the job runs and pays (transfers currency) to the others based on their *expected* tolerance values (i.e. the mean $t_i \cdot d_j$ over the distribution of parameter values for the task). If the value v_i is lower, then the job is dropped without transfer of currency. The user may choose to resubmit the job, possibly with different v and d values. (Note that while the job was waiting in the queue, it acquired currency from the jobs that ran before it; this would allow it truthfully to claim a higher value, for example.) This pricing mechanism has two useful theoretical properties. Truth-telling (i.e. accurately setting t, d, v values) is a Nash equilibrium strategy; that is, if all other users are telling the truth, one user cannot improve her situation by providing false information. Also, the currency budget is balanced; that is, payments from the executed jobs equal the accruals by the waiting jobs. This implies that the system administrators can set user priorities simply by their initial distribution of currency. The graphs below, reproduced from the paper, demonstrate the former property in a simulation experiment. One simulated user gave inaccurate parameters for job value (left graph) or delay tolerance (right graph). The red dots track the outcome (actual value of work) for the users; black dots are individual simulations that create these averages. Note that the best outcome – i.e. the highest average final value – occurs when the parameter is correct.



In unpublished (to date) work, Mutz has also explored another approach to economics-based scheduling based on auctioning reservation slots. In this work, the user supplies the job value (v), run time (t), and deadline (d). Dynamic programming can then construct a provably optimal (with respect to value of executed jobs) schedule. Mutz has integrated this method with a PBS queue system, computing the reservations once

per day. Like the expected externality method, users get the best outcomes when they announce accurate parameters for their jobs. The graphs below show this.

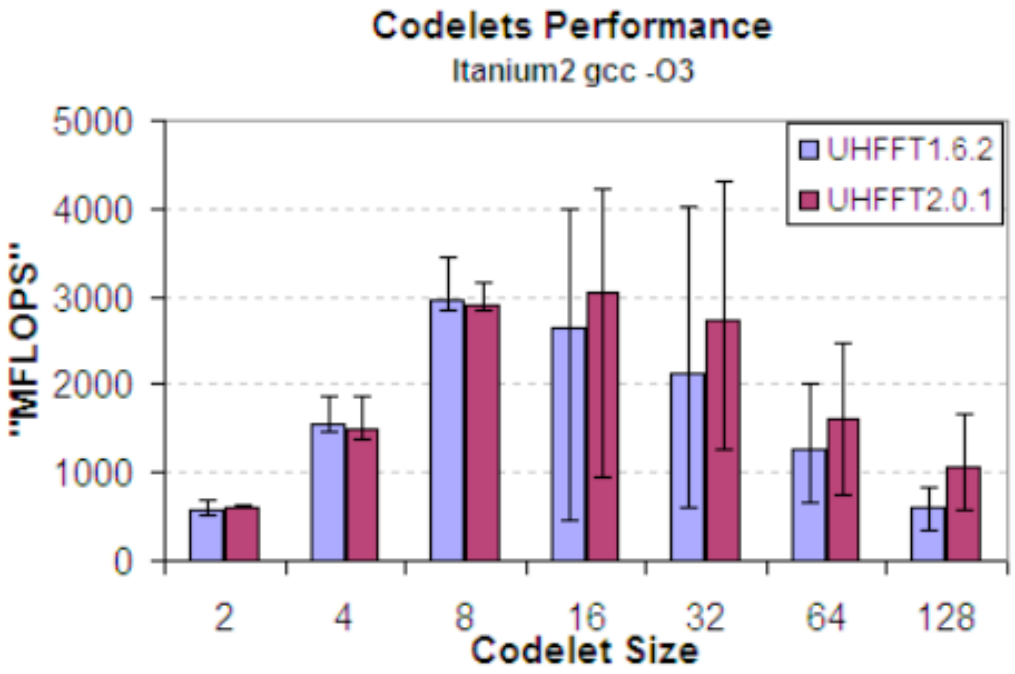


2.3 Compiling and Optimizing Node Programs

In past reports we have discussed activities at Rice University to use Just-in-Time (JIT) compilers to improve the efficiency of node programs. As the student involved in those efforts graduated, the node compiler activity moved to the University of Houston.

Ayaz Ali, a student of VGrADS PI Lennart Johnsson, has published a series of papers about compilation of the Fast Fourier Transform algorithm for modern computer architectures. Because the FFT uses very particular data access patterns and non-nearest-neighbor communications, many compiler techniques developed for linear algebra and mesh relaxation do not produce good results. However, the variety of implementation options available (corresponding to recombinations of the Kroenecker product formulation of the FFT) allows many opportunities for optimization. In "Empirical Auto-tuning Code Generator for FFT and Trigonometric Transforms" Ali, Johnsson, and Dragan Mirkovic present a code generator framework for exploiting these opportunities. Their three-stage scheme first generates *codelets* (straight-line program blocks) for the basic FFT butterfly operation, then schedules the order of the codelets by considering them as a DAG computation, and finally chooses the formats

for the input and output arrays to produce the best running time. Each stage selects appropriate parameters for the code it generates using a heuristic search strategy, in which possible parameters are evaluated by empirically testing the code. As the graph below (taken from the paper) shows, this can result in significant improvements over an older strategy based only on heuristically choosing few parameters without empirical testing.

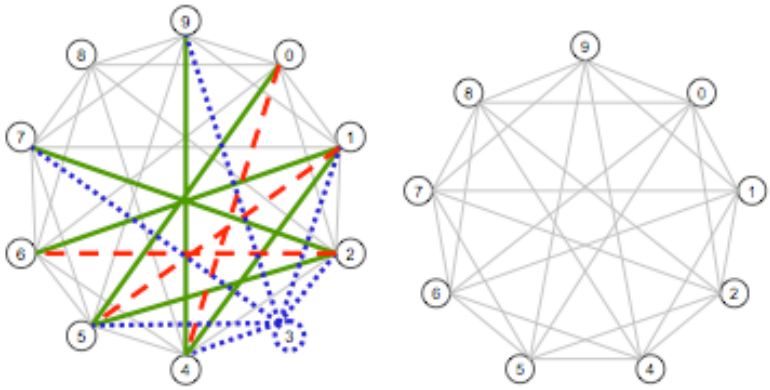


2.4 Fault tolerant MPI: FT-MPI / OpenMPI

Fault Tolerant MPI (FT-MPI) is a full MPI 1.2 specification implementation developed at UTK that provides process-level fault tolerance at the MPI API level. It is one of several fault tolerance techniques being incorporated into the OpenMPI project. This umbrella project 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. VGrADS has supported this work (in cooperation with other grants) since its inception.

During the past year, VGrADS PI Jack Dongarra led a team including Thara Angskun, George Bosilca, Graham Fagg, and Jelena Pješčivac–Grbović in investigations of self-healing network topologies. In a series of papers, they have investigated the fault tolerance properties of networks through discrete event simulation (rather than the more common approach of combinatorial analysis). For example, in "Self-Healing in Binomial Graph Networks" they study the connectivity of the family of binomial graphs

(those in which each node n is connected to nodes $(n \pm 2^k)$ *MOD* m for all $2k<m$ where m is the size of the graph). They find that the graph is δ-connected, where δ is the degree of each node, and that the average number of disconnected nodes and average hop count increase significantly when the number of node failures rises to 50% of the graph size. However, by using the efficient methods for self-healing the binomial graph (which may be triggered automatically, it is possible to create a new, smaller binomial graph. Illustration (a) below shows how this is done on a 10-node binomial graph when node 3 fails. First the links to node 3 (blue dotted lines) are eliminated; then other superfluous links (red dashed lines) are removed; finally, new links (green solid lines) are added to rebuild the topology. The resulting graph – the binomial graph of size 9 – is shown in illustration (b).



A significant contribution of the paper is showing how these links can be identified locally, rather than causing a global synchronization. Dongarra and his team are investigating incorporating these fault tolerance methods in a future implementation of FT-MPI and OpenMPI.

3 VGrADS Execution System (UCSB, UCSD, UNC, USC/ISI)

As in past reports, our execution system research directions address the continuing development of vgES (Virtual Grid Execution System) to support complex, adaptive workflow applications. Section 3.1 describes the basic capabilities of vgES, which might be considered the software infrastructure for VGrADS. Other sections expand on research that extends parts of this system.

While basic development and improvement of vgES has occurred, much of our effort is now directed at integrating "cloud computing" resources into vgES and the VGrADS testbed. Section 3.2 describes this new research thrust. Such support will be the focus of the No Cost Extension that was recently granted.

We have also refined our support for temporal reasoning for resource characterization. This work appears in full in Emma Buneci's thesis, and in greatly abbreviated form in Section 3.3.

In addition to the above, we have conducted research in the four areas described in detail in the indicated sections: predication based methods for virtual reservations (Section 3.4), algorithms for optimal slot selection (Section 3.5), fault tolerance in the Virtual Grid (Section 3.6), and Grid resource specification generation (Section 3.7).

As it has been for the past several years, vgES formed the basis for a VGrADS demonstration at the annual SC conference (SC07 in Reno). Multiple members of the execution system team presented these demonstrations at the GCAS (Gulf Coast Academic Supercomputing, a consortium including Rice and UH), SDSC, and RENCI booths. The innovation this year was the addition of fault tolerance to the end-to-end demonstration. While we had expected (due to time constraints) to only be able to show this as a simple movie of a previous experiment, fate intervened. During our live demonstrations, various glitches (including a city-wide power outage) created actual failures in the presentation. However, since these only affected the local area, our own grid software withstood the challenge.

3.1 vgES Fundamentals

As it has been from the beginning of the project, vgES provides VGrADS with a framework for resource management and an execution environment for grid applications. Here, we briefly describe the current core concepts of that framework.

3.1.1 A Generic Framework for Resource Allocation and Execution

Since many Grid resource managers were designed for time-sharing or dedicated resources, the resource acquisition process has been ignored or, at most, been simple and naive. In the real world, however, most resource providers employ a resource

manager for efficient utilization and better services, which makes the resource acquisition process complex. As a solution, we implemented a new resource acquisition mechanism called *resource actualization* that consists of *orchestration* to make a resource bound and *personalization* to configure the bound resources with appropriate execution environments.

Resource orchestration coordinates distributed resources and transparently acquires a resource collection for a resource specification, isolating the user from the heterogeneity and dynamics of the underlying resources. In essence, it defines the meanings and the mechanisms of binding a resource collection for a specification. A resource collection is called *bound* when it is made available. vgES orchestrates multiple resources simultaneously against binding failures and determines the best one among the bound resources, considering the characteristics of application and resource.

Resource personalization implicitly configures the bound resources with an execution environment, based on the application characteristics. In essence, resource personalization leverages commodity tools for task scheduling, resource management, communication, etc, to simplify application development and exploit the features of the tools. A fundamental difference from conventional approaches such as Plush and Condor-G, which configure a predetermined execution environment, is that resource personalization enables arbitrary coupling of resources and execution environment. Moreover, any commodity tools can be plugged in as long as they conform to the external APIs for extension.

3.1.2 Slot Allocation over Advance Reservation and Virtual Reservation

Initially, vgES 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. This was inadequate for real applications, such as the LEAD workflow (see Section 1.1). Its requirements are indicative of the complicated machinations required for adaptive, real-time applications. We addressed this shortcoming by refining the time notion in vgES to accommodate batch queues. The resulting construct was termed a *slot*.

A slot is a high level representation of resources in time and space. It is a set of resources available in a certain time range. It can be expressed by a tuple <size, start time, duration>. Different from the slot used in a batch system that describes one resource allocation across time and space for the single resource from the system viewpoint, the slot in vgES is a resource collection in time and space across multiple resources from the application viewpoint.

We implemented this slot concept against advance reservations provided by some batch systems and virtual reservation using batch queue prediction (see Section 3.4). Further, we are working on allowing the users to apply operations to slots to set slot properties

(e.g., dependency between slots) or to change time constraints (e.g., modify start time and/or duration).

3.1.3 Resource Equivalence for Flexible Resource Discovery

Users can experience resource discovery failures due to a variety of causes such as resource scarcity or resource conflicts. With these search failures, users can just repeat the same request until the resources are available or modify their requests, anticipating successful search. As an alternative, the system can provide more flexibility in resource discovery to improve the likeliness of search success. A popular feature that the traditional resource brokering systems provide is a range search. Compared to exact matching techniques, it can explore more candidates.

However, range search provides flexibility to only a single attribute. To provide more flexibility across attributes, we extended the vgDL language to enable the users to find the resources successfully even in case of discovery failures. The user can define a set of equivalent resource configurations that have precedence and expect to achieve similar performance on a per-application basis. For instance, the developers can specify that an Opteron processor achieves three times better performance than an Itanium 2 processor for a given application as follows: Opteron <> 3 * Itaniutm2. Resource equivalence as well as range search is reactive to search failures and makes the system resilient to the discovery failures. Both of them increase the number of resource candidates in the resource pool and consequently improve the likeliness of successful research discovery. This extension introduces another dimension in resource selection, which makes the selection more complex. We are exploring the tradeoff between resource quality and selection cost.

3.2 Cloud Computing Local Cluster Resources

During the reporting period, the UCSB team headed by Dan Nurmi and Graziano Obertelli began developing EUCALYPTUS – An Elastic Utility Computing Architecture for Linking Your Programs to Useful Systems. Eucalyptus is an open-source reverse-engineered implementation of Amazon's EC2 cloud computing infrastructure that can be deployed on local cluster resources. Using SDSC Rocks as a deployment tool, any site installing Eucalyptus can offer cloud-computing services from its resources using the user-interface and command-line tools available from Amazon.com. It is EC2@home.

The intention is to include cloud computing in the portfolio of systems that vgES can amalgamate automatically beneath the VGrADS slot abstraction. Various VGrADS sites (UCSB, RENCI, Rice, UH) will run Eucalyptus, each exporting its own cloud, which will then be integrated into a single set of slots by vgES. In addition, because Eucalyptus uses the Amazon.com interface tools, vgES will be able to use EC2 itself (although there will be an occupancy charge). The project intends to explore how

commercial "for-fee" cloud computing services (EC2), collaborative locally installed cloud-computing services (Eucalyptus), and NSF center services (TeraGrid) can be profitably combined for workflow applications via the slot abstraction.

3.3 Qualitative Performance Analysis Framework For Scientific Workflows

We have continued the development of a qualitative temporal reasoning framework to support performance validation and diagnosis of long-running scientific Grid applications for Virtual Grids (VGs). Our goal is to reason about temporal events to differentiate between severe, persistent behavioral violations and transient ones, and to output a qualitative answer to a workflow user or fault-tolerance service. The framework's qualitative performance output can help bind the expectations of applications with the resource behavior in the VG execution system.

The qualitative performance analysis framework supports Grid environments in (a) monitoring and validating the temporal behavior of long-running scientific workloads, (b) diagnosing possible sources of behavior changes, and (c) providing reasoning support for applications to adapt to behaviors associated with degraded application performance.

3.3.1 Framework Overview

We have developed a novel performance analysis framework that reasons temporally and qualitatively about performance data from multiple monitoring levels and sources. The framework periodically analyzes application performance states by generating and interpreting signatures containing structural and temporal features from time-series data. Signatures are compared to expected behaviors and, in case of mismatches, the framework hints at causes of degraded performance, based on unexpected behavior characteristics previously learned by application exposure to known performance stress factors.

3.3.2 Progress

In the reporting period (June 2007 – May 2008), we have focused on expanding experimental validation to more Grid computing resources and applications, refining algorithms and methodologies used, and validating the efficacy of our framework on the collected experimental data.

We have expanded experiments with two Grid workflows from meteorology (LEAD) and astronomy (Montage). We have run long-running tasks within these workflows with two different input data sets, and on four different Grid computational resources (NCSA TeraGrid Mercury clusters and RENCI's Dante cluster). Performance monitoring data captured during these executions reveal common qualitative temporal signatures characterizing successful and well-performing execution for both

applications. This allows the framework to learn and store the signatures of typical, healthy temporal behaviors. Furthermore, we collected and analyzed temporal signatures generated from performance data captured during known degraded-performance application states. The resulting signatures are significantly different from the expected, healthy behavioral signatures previously learned during well-performing application executions. The ability to automatically and compactly generate signatures capturing fundamental differences between good and poor application performance states is essential to improving the quality of service for Grid applications.

We have refined the algorithms used for time series pattern recognition, employing an improved heuristic based on a combination of statistical hypothesis testing with properties of the autocorrelation function. Also, we have studied several approaches for a better visualization of the multi-dimensional data contained in the behavioral signatures generated. We have employed star-coordinates as a technique to support and guide the analysis and characteristics of groups of temporal signatures, and we have also developed a visualization of temporal signatures based on coloring of the vector values of the signature.

We have evaluated the efficacy of our framework based on the primary methodology of use: classification of new temporal signatures generated from on-line performance monitoring data. We have employed the balanced accuracy as a measure of the ability to accurately identify both signatures from the expected performance states and from the unexpected/diagnostic performance states. Our results indicate that our qualitative framework has the ability to identify correctly performance problem states over time with accuracies ranging from [61% - 99%], depending on the application analyzed and various transformations applied to the performance data. The obtained accuracies are sufficient to indicate potential problems during the execution of long-running workflows, especially considering the very small temporal signature and analysis overhead incurred by our qualitative performance analysis framework.

3.4 Predication Based Methods for Virtual Reservations

The slot based vgES can construct a virtual grid from service level agreements (i.e. advanced reservations) or, in the situation that advanced reservations are not supported by a resource provider, it can exploit predictions of future resource availability, or virtual reservations. To this end, we developed VARQ -- Virtual Advanced Reservations for Queues -- to support the vgES slot abstraction in best-effort batch environments. The vgES slot abstraction allows vgES clients to specify when a specific resource set should be made available to them, and for how long. In environments where advanced reservations are supported, vgES simply uses the local reservation infrastructure. In the majority of grid settings VGrADS is currently targeting, however, no advanced reservation capability is supported. On these systems, VARQ implements a "virtual" reservation by making statistical predictions of the queue delay associated

with different possible submissions. It then chooses the submission time and format most likely to provision vgES by the reservation time. The VARQ implementation is in the form of a library that has been bundled together with vgES and is part of any standard vgES installation.

VARQ depends critically on another VGrADS contribution: the QBETS system. QBETS (Queue Bounds Estimation from Time Series) uses a new time series prediction method to estimate the instantaneous bounds on queue delay an individual job will experience. Using newly developed Network Weather Service queue delay monitors (also a VGrADS-funded innovation), QBETS constantly monitors the delay response exhibited by a target machine. As jobs pass through the system, it identifies and updates the historical information necessary to make predictions for individual jobs. In particular, it automatically identifies change-points in each historical series (so that only relevant data is considered) and model-based clustering to categorize jobs into service classes. The effect is an instantaneous prediction that automatically takes into account service-class specific priorities such as those implied by back filling. Together, VARQ and QBETS provide vgES and the HPC community as a whole with the ability to specify start time deadlines for jobs requesting time on batch controlled resources: a new and important functionality for the Grid computing community.

VARQ, using QBETS, successfully provided virtual advance reservations on VGrADS site machines (SDSC's ia64 TeraGrid Cluster, Rice's RTC Cluster, RENCI's Dante Cluster, and UCSB's Mayhem Cluster) to vgES as part of the LEAD VGrADS demonstrations every year since SC06. It is also analyzed in several papers listed in the bibliography and in submission. QBETS research has resulted in infrastructure (not supported by VGrADS) that is currently deployed across the TeraGrid as a user advisory tool and also as part of the TeraGrid user portal.

3.5 Algorithms for Optimal Slot Selection

One of the next steps in the evolution of vgES is to enable it to choose intelligently between alternative slots when implementing a VG. Such selection requires that vgES be able to balance the costs and benefits of alternative resource providers and take into account the fact that the cost of an advanced reservation will not be uniform across all resource providers. The work on application-level resource provisioning addresses these issues and describes a framework for resource allocation and scheduling in provisioning based systems. We model the resource availability as a set of resource slots where each slot implies the availability of certain resource capability for a certain timeframe for a certain price. In a real system, a slot can be implemented as a reservation. Given this resource model, the problem we solve is to select a set of slots, called the resource plan, for the application that optimizes the application performance while minimizing the resource costs. The problem is difficult to solve exactly because of the large number of feasible resource plans that could be used to execute the application. Thus we use heuristics such as Multi-Objective Genetic Algorithms to

search the solution space and create a small set of potential candidates. Then a user specified preference factor is used to select one solution from these candidates. The application performance that we seek to optimize is the completion time of the application, also known as the *makespan*. In addition to the resource plan, the makespan also depends on the heuristic for scheduling the application over the slots in the resource plan. In this work, we use the well-known Heterogeneous Earliest Finish Time (HEFT) scheduling algorithm.

3.6 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. 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, this year we implemented the fault tolerance extensions to the virtual grid description language. In addition, we are using the idea of performability as a metric for resource selection and workflow scheduling. Performability is the joint treatment of performance and reliability introduced by Meyer in 1978. Performability provides a composite measure of a system's performance and reliability and gives the system a chance to qualify system performance in the event of failures. Performability analysis has been applied to computer networks and communication systems, but has not been applied to higher-level workflow representation and/or programming models.

We conducted an experimental evaluation of the system with two application examples: a) Balancing performance and reliability in scheduling workflow components: A multi-level scheduling approach uses reliability as a criterion for resource selection in addition to performance. In addition, based on the priority of workflow tasks, parts of the workflow are replicated appropriately. b) Multi-level fault tolerance mechanism at the vgES and workflow level: We use a simple cost model to balance workflow replication with checkpoint frequency to demonstrate the choices applicable based on different programming models.

Our current activities include comparison of two prevalent fault-tolerance mechanisms—over-provisioning and simple restart (simple form of migration) in terms of time taken to successful completion in the presence of failures. We are also working on algorithms to determine (a) the degree of over-provisioning: how many copies of the application need to be run on which resources to ensure that at least one copy succeeds (with a given probability) within the deadline, and (b) migration path: determining to which resources to migrate in case of a failure such that the application succeeds (with a

given probability) within the deadline. We consider the reliability (one-hour failure probability) of the underlying resources and the performance of the application on the resources to determine the suitable degree of over-provisioning/migration path.

In particular, we consider the following information about the available resources and the application to estimate the expected performance of the application: (a) latency and bandwidth information from NWS and data sizes from data-sources to estimate expected communication time, (b) BQP (Batch Queue Prediction) data to estimate the time the application has to wait on a queue on a particular resource before it can start execution, (c) whether there is a reservation on a queue on a resource, (d) MDS information to find out available resources and queues, and (e) computational performance estimates of the application in the form of performance models (table look-up). We are working on developing a method for doing automatic trade-off between over-provisioning and migration.

3.7 Grid Resource Specification Generation

Any resource selection system (such as vgES) requires as input a resource specification (vgDL for vgES) describing the desired set of resources to execute the application. However, application developers are often focused on optimizing the application and application users often do not have the insight into the application to produce a resource specification that can optimize the application performance. While we have shown that an appropriate virtual grid (VG) produced by vgES can both optimize application performance and simplify scheduling, what is not clear is how to generate the resource specification that can produce the appropriate virtual grid. To address this issue, we formulated an empirical model to generate resource specifications based on application characteristics, the scheduling heuristic used, and an optional utility function allowing users to trade off application performance and resource cost. We validate our model with Montage DAGs and with arbitrarily generated DAGs. Our validation results show that our model leads to VGs enabling good performance at low resource costs. Further, we validate that our model maintains good performance for different scheduling heuristics and for different resource heterogeneity within the resource universe. The framework of our model appeared at HPDC 2007.

4 Management & Structure

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/ISI); and University of Tennessee, Knoxville (UTK). Rice University serves as the lead VGrADS institution. Keith Cooper serves as VGrADS PI. He is advised by an executive committee, which consists of the key researchers leading the main VGrADS research thrusts. The current members of the VGrADS executive committee are:

Keith Cooper (Rice, Chair)
Jack Dongarra (UTK)
Carl Kesselman (USC/ISI)
Chuck Koelbel (Rice)
Rich Wolski (UCSB)

During the reporting period, the VGrADS executive committee met on a regular basis either in person or by teleconference to review progress and milestones, discuss plans for the future, and advise the PI on resource allocation issues.

Project design and coordination during the current reporting period were enabled through weekly technical teleconferences involving researchers from each of the VGrADS sites, one PI meeting, regular VGrADS executive committee meetings and teleconferences, VGrADS planning workshops at UNC (10/11-12/07) and Rice (4/7-8/08), one developers' workshop at USC/ISI (8/1-2/07), and communication via VGrADS mailing lists. Research subproject participants also met on a regular basis to exchange ideas and develop research plans. In addition, the EOT group at Rice met to develop and coordinate education and outreach activities.

4.1 Changes to Management Structure

On December 3, 2007, VGrADS PI Dan Reed left UNC to join Microsoft as Director of Scalable Computing and Multicore (http://www.renci.org/news/transition.php). Anirban Mandal, who has been a key participant throughout the VGrADS project, assumed the role of lead PI at UNC.

4.2 VGrADS Web Site

During the funding period, the VGrADS Web site, http://vgrads.rice.edu, was updated to reflect recent results, current project directions, and personnel changes.

5 Project Milestones

As recommended by the NSF Site Review team (4/28-29/05), the VGrADS Principal Investigators have actively tracked and (where necessary) updated research milestones for the project. Here we report on milestones in years 4 (some of which were still in progress as of our last report), 5 and 6. All section numbers refer to the 2008 report unless otherwise noted.

5.1 Year 4 Milestones

Our revised milestones, and the progress toward them, included:

i. Execution System/Virtualization:
   - Evaluate resource selection, binding, and scheduling techniques based on time-dependent provisioning for VGs developed in Year 3.
     (Done: described in 2007 report.)
   - Explore techniques to extend the VG abstraction to diverse resource management paradigms (e.g., probabilistic space-time resource abstractions).
     (Done: described in 2007 report and in Section 3.1.)
   - Expand techniques for integrated time-space reasoning with performance/fault tolerance capabilities.
     (*Done: described in Section 3.3.)*

ii. Execution System/Grid Economy:
   - Complete integration of new prediction capabilities with vgES to support both scheduling and grid economy work.
     (Done described in 2007 report, described in Sections 2.2 and 3.4.)
   - Conduct empirical investigation of pricing scheme and its effect on resource allocations.
     (Done: described in Section 2 of 2008 report.)

iii. Execution System/Fault Tolerance:
   - Implement novel techniques (e.g. diskless checkpointing in a general setting in Open-MPI).
     (Done: described in 2007 report and in Section 2.4.)
   - Prototype and experiment with techniques to implement dynamic adaptation in a multi-level fault tolerance environment on the virtual grid.
     (Done: described in 2007 report and in Section 3.6.)

iv. Programming Tools/Workflow:
   - Demonstrate novel, scalable workflow scheduler(s) on TIGRE grid.
     (Partly done: Schedulers demonstrated (see Section 2.1), but TIGRE grid is only partly compatible with VGrADS requirements.)
   - Explore robustness of workflow schedulers, particularly with regard to fault tolerance.
     (Done: described in 2007 report.)

v. Applications:
   - Incorporate novel, scalable workflow scheduler(s) into EMAN and LEAD applications.
     (Done: described in 2007 report and in Section 1.1.)
   - Continue evaluation of fault tolerant techniques with LEAD.
     (Done: described in 2007 report.)

vi. Education, Outreach, and Training:
   - Continue AGEP program.
     (Done: One student supported in summer 2007 (expected second student dropped out).)
   - Participate in Tapia Symposium.
     (Done: VGrADS co-PI was poster chair, will support student travel.)
   - Continue graduate student exchanges and collaborative workshops.
     (Done: collaboration continues.)
   - Participate in Grace Hopper Conference.
     (Done: supported student travel.)

5.2 Year 5 Milestones

We have made excellent progress on Year 5 milestones. We expect to achieve the following milestones by the end of the award period:

i. Execution System/Virtualization:
   - Improve resource selection and binding techniques for flexible resource discovery.
     (Done: described in Section 3.7 of 2008 report. Some related work also described in 2007 report.)
   - Improve resource-scheduling techniques for VGs that consider resource efficiency and cost.
     (Done: described in Section 2.1 and 2.2. Some related work also described in 2007 report.)
   - Prototype and evaluate extended VG abstraction over environments with diverse resource management paradigms.
     (Done/in progress: described in Section 3.1.3 and 3.2. Some related work demonstrated as early as SC06.)
   - Demonstrate vgES with resource selection, resource binding, VG scheduling, for application kernels across large-scale grid platforms with diverse resource management paradigms.
     (Done: described Section 3.1, 3.2 and 3.4. Also referred to in Section 1.1 as a key part of our LEAD demo.)
   - Validate and assess the integrated resource provisioning policies.
     (Done/in progress: Work started in Sections 3.5 and 3.7.)

ii. Execution System/Grid Economy:
   - Investigate adaptive pricing algorithms for resource reservations that accurately reflect their value when compared with typical best-effort

queueing service.
(Done: described in Section 2.2.)

- Design experiment to investigate allocation efficiency under various pricing schemes.
  (Done: described in 2007 work and Section 2.2.)
- Target second VGrADS-enabled application (to be determined) as a driving application.
  (Done: described in 2007 report.)
- Verify using both GridSAT and second application.
  (Done: GridSAT and LEAD described in 2007 report.)

iii. Execution System/Fault Tolerance:

- Integrate fault tolerance features into vgES and test on LEAD application.
  (Done/in progress: described in Sections 1.1 and 3.6. Improvements continue, and will be demonstrated at SC08.)

iv. Programming Tools/Workflow:

- Incorporate novel techniques from vgES and fault tolerance work into workflow schedulers.
  (Done: described in Section 2.1.)
- Study new problems in workflow based on grid economies, fault tolerance, and dynamic resource behavior.
  (*Done/in progress: scheduling for fault tolerance described in part in Section 2.1 (though work continues). Dynamic resource behavior is a significant part of the cloud computing work described in Section 3.2.)*
- Consider compilation approaches to task migration for fault tolerance.
  (In progress. Preliminary work is part of the scheduler research described in Section 2.1.)

v. Applications:

- Evaluate methods for scheduling workflow applications on large-scale TIGRE grid.
  (Partially done: TIGRE grid project diverged from VGrADS infrastructure.)
- Explore additional TIGRE applications.
  (Not done: TIGRE grid project diverged from VGrADS infrastructure.)

vi. Education, Outreach, and Training:

- Continue AGEP program.
  (In progress: two students started AGEP summer program.)
- Continue graduate student exchanges.
  (Continuing effort.)

5.3 Year 6 Milestones

As part of our requested No Cost Extension, we developed a set of milestones for finishing the project. Most of these are extensions of Year 5 milestones, specialized based on our current development:

i. Execution System/Virtualization:
   - Extend vgES to manage dynamic "cloud computing" resources.
     (In progress: first steps described in Section 3.2 will be demonstrated at SC08.)
   - Investigate scheduling methods for systems with both allocated clusters and cloud computing resources.
     (In progress: first steps described in Sections 3.1 and 3.2 will be demonstrated at SC08.)

ii. Execution System/Grid Economy:
   - Continue investigation of grid economy systems for resource allocation and scheduling.
     (In progress.)

iii. Execution System/Fault Tolerance:
   - Continue investigation of fault tolerance measures for cloud computing.
     (In progress.)

iv. Programming Tools/Workflow:
   - Demonstrate workflow schedulers sensitive to fault tolerance and performance model considerations.
     (In progress.)

v. Applications:
   - No additional milestones expected.

vi. Education, Outreach, and Training:
   - Support 2009 Richard Tapia Celebration of Diversity in Computing.
     (In progress: VGrADS PI Koelbel is on Tapia2009 committee, and we will support student attendees if funds permit.)

II. Findings

During the reporting period (6/1/07–5/31/08), VGrADS research continued to focus on three inter-institutional efforts: *Applications, VGrADS Programming Tools,* and *VGrADS Execution System*. The following sections summarize the findings of each subproject.

6 Applications

LEAD: We successfully demonstrated the applicability of the VG abstraction and VGrADS scheduling and fault-tolerance techniques to the workflow from an important meteorological application. This can (in principle) address LEAD's requirement for transparent resource selection, monitoring and runtime adaptation. We have verified that our fault-tolerant scheduling techniques provide a means to address LEAD reliability and quality of service requirements.

EMAN: We demonstrated how a workflow application could be effectively scheduled to use multiple clusters, each managed by an independent batch queue. We showed how this scheduling could lead to substantial improvements in turn-around time.

GridSolve: We demonstrated a flexible, user-friendly interface to important mathematical software accessed over the grid. This included resource discovery, scheduling, and load balancing. We are currently adapting the software to take advantage of the VG abstraction through vgES.

7 VGrADS Programming Tools

Scheduling Workflow DAGs: We have shown that the two-phase scheduling strategy of choosing a VG, then scheduling to the resources in that VG, gives good results on a variety of real-world and randomly-generated DAGs. Moreover, we have found that the reduction in grid size that VG selection provides makes it feasible to use more advanced scheduling heuristics, thus producing improved schedules. We have demonstrated that list-based scheduling algorithms produce excellent results, particularly when coupled with selection of clusters using the estimated aggregate computing power.

Scheduling Applications onto Batch Queues: We have shown that accurate predictions of batch queue wait time are possible (albeit with unavoidably large error bounds in some cases). We have used these predictions in conjunction with predictions of computation and communication time to schedule the EMAN application. This scheduling mechanism produced integer factor improvements in turn-around time.

Scheduling for Reliability: We have shown how applications can trade off reliability (probability of successful completion) and performance by a novel scheduling

algorithm. The insight of the algorithm is that using resources with minimal {execution time}×{failure rate} maximizes reliability for any given makespan. Therefore, ordering processors by this quantity when choosing resources to use allows the application to optimize its reliability for a given deadline (or its execution time for a given reliability). This method can be applied to a variety of schedulers.
Resource Economies: We find that computationally efficient algorithms for "solving" GVA allocation problems can be developed and used to design novel reservation protocols for batch-controlled systems. These protocols are incentive compatible (truth-revelation about resource requirements is a dominant strategy) and budget-balanced. They can also be implemented using existing open-source tools such as PBS.

8 VGrADS Execution System

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 real-time applications, which allocate resources against advance reservation and best-effort batch resources. Moreover, further studies on the resource actualization process enable us to redesign vgES as a generic framework for resource management and execution environment. The detailed resource instantiation mechanisms have been studied and submitted to Grid'07.

Flexibility in Discovery: We extended the vgDL to express resource equivalence. This extension provides more flexibility in resource selection, which eliminates the iteration of resource selection in cases where the selection operation fails. The preliminary implementation, which allows the user to specify equivalence for processor type, can discover equivalent resources with a small additional overhead of 5 -10%.

Resource Provisioning: Our work with optimal provisioning has shown that resource provisioning generally leads to a better application performance than best-effort service for applications with large resource requirements and when systems are under high utilization.

Virtual Resource Reservations: We find that VARQ is able to "manufacture" a probabilistic resource reservation on systems that only support best-effort batch queue service. By predicting when a job should be submitted in the future in order to meet a specified deadline, the system ensures that the user will be guaranteed the resources during the desired timeframe (i.e. has a reservation). We find that VARQ reservations are often a preferable substitute for "hard" advanced reservations since the latter must currently be negotiated manually by members of our research team and the site administrators controlling the machines we currently target (e.g. the TeraGrid resources). Lastly, the Slotted Virtual Grid abstraction is essentially an abstract reservation that is translated directly (with little alteration) into a hard reservation once it is made by hand. We find that a VARQ reservation can support the SVG abstraction

fully automatically with no intervention by the user or relevant system administrators, and can do so in production grid environments.

Fault Tolerance: Our work on fault tolerance has shown that performance and reliability models enable automatic selection of over-provisioning, migration, or restart options that hide the details of grid service failures while maintaining the virtual grid abstraction.

Cloud Computing: We find that cloud computing services can be implemented using cluster resources locally procured and maintained for scientific research (as opposed to being purchased exclusively from "for fee" service providers). We also find that both commercial and Eucalyptus-supported services can be integrated to support the VGrADS slot abstraction by the vgES.

Resource Economies: We find that computationally efficient algorithms for "solving" GVA allocation problems can be developed and used to design novel reservation protocols for batch-controlled systems. These protocols are incentive compatible (truth-revelation about resource requirements is a dominant strategy) and budget-balanced. They can also be implemented using existing open-source tools such as PBS.

Resource Specification Generation: We constructed an empirical model for resource specification generation that enables vgES to return an appropriate VG, leading to good application performance for arbitrary applications expressed as Directed Acyclic Graphs (DAGs).

III. VGrADS Education, Outreach, and Training Activities

The following sections describe VGrADS Education, Outreach, and Training (EOT) activities during the current reporting period (6/1/07-5/31/08).

1 Training and Development Activities

Much of the 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.

An important part of this interaction was student attendance at small-group workshops, which we call developer workshops. The demonstrations (at SC06 and SC07) of end-to-end VGrADS capabilities led us to emphasize these meetings rather than student exchanges as we did in early years of the project. The developer workshops are working meetings, producing detailed plans and software artifacts for use in our experiments and demonstrations. This year, we held one such meeting at ISI on August 1-2, 2007 to integrate virtual grid development, fault tolerance, and scheduling on the LEAD application. The students involved received significant experience in collaborative work and distributed software development, as well as a broader exposure to the project than they would ordinarily have had. Of course, the developer workshop also had a great positive effect on the demonstrations described elsewhere in this report.

1.2 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.

1.3 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 2008 semester, Lennart Johnsson taught *COSC 6365: Introduction to High Performance Computing* at UH. For more information on this course and its contents, visit either the course announcement page at http://www2.cs.uh.edu/~johnsson/cosc6365_08/Course_Announcement.pdf or the course Web page at http://www2.cs.uh.edu/~johnsson/cosc6365_08/. In the Spring 2008 semester, Jack Dongarra taught *CS 594-004: Scientific Computing for Engineers* at UTK, which covered current trends in high-end computing systems and environments, parallel programming, aspects of Grid computing, and other topics relevant to scientific computing. For more information on this course and its contents, visit http://www.cs.utk.edu/%7Edongarra/WEB-PAGES/cs594-2008.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 having VGrADS researchers serve as mentors for these summer students. AGEP leverages VGrADS to reach more students, through our direct funding of additional participants.

In summer 2007, we sponsored one female student, Emily de la Garza, who had also participated in 2006. Under the direction of VGrADS co-PI Chuck Koelbel, Emily, now a senior in the computer science program at the University of Houston-Downtown, used the EMAN program to continue her work on comparing and contrasting the use of batch queues with performance models and Condor. She also reports using her experience

with VGrADS to help set up a grid testbed at UH-D, although that research is independent of the VGrADS project *per se*. (Another potential summer student, Lauren Garcia, was mentioned in the 2007 annual report; unfortunately, she had to withdraw from the VGrADS portion of the summer program before the program started, and we were unable to fill her slot.)

We have also recruited participants for summer 2008, which is getting underway as this report is being submitted. Out of many applicants, we selected Keisha Cumber (from Johnson C. Smith University in Charlotte, NC) and Stephanie Diehl (from Case Western University). Both are freshmen. They will be working under the joint mentoring of Chuck Koelbel and Vivek Sarkar (a non-VGrADS-affiliated professor of CS) on projects related to parallel and distributed computing. We have supplied both students with beginning textbooks to bring them up to speed on the topic. 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. In 2007, VGrADS was able to provide travel support for one VGrADS summer student to attend the Richard Tapia Celebration of Diversity in Computing Conference 2007 (http://tapiaconference.org/2007/).

The Tapia conference is a biannual meeting; the Hopper conference is now an annual meeting. The Grace Hopper Celebration of Women in Computing 2007 (http://gracehopper.org/2007/) was held October 17-20, 2007, and was co-located in Orlando, Florida with the Richard Tapia Celebration of Diversity in Computing Conference 2007 (http://tapiaconference.org/2007/), which was held October 14-17, 2007. The two conferences shared a day when attendees from both conferences attended joint sessions. Dr. Charles Koelbel, a VGrADS PI, served as Posters Chair for the Tapia Conference 2007 and will be doing so again in 2009.

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

VGrADS PIs have continued to participate in the NSF-funded Computer Science Computer and Mentoring Partnership (CS-CAMP) project (http://ceee.rice.edu/cs-camp/). VGrADS PIs Keith Cooper and Richard Tapia are PIs of the CS-CAMP project. MS CS-CAMP, an extension of the original CS-CAMP program for high school girls, is designed for middle school girls. The first session of the program, which served 44

middle school girls during the summer of 2007, will be followed this summer (2008) with another session. At least 35 girls are currently registered for that camp.

MS CS-CAMP is a one-week version for middle school girls of the successful two-week CS-CAMP program for high school females. Both programs are designed to encourage and motivate females to think about computer science as a possible career choice, and to arm them with skills to help them succeed in high school computer science classes. The students participated in a wide variety of sessions related to computer science, involving robotics, programming in Scheme, logic, and – of course – grid computing.

VGrADS PIs Keith Cooper, Richard Tapia, and Chuck Koelbel, who were involved in planning the new program, participated in sessions during the first middle school camp, and will do so again during the summer of 2008.

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 SC07 Conference (Reno, NV, November 10-16, 2007). Other sections of this report detail the research advances that were reported there, including paper and poster presentations. VGrADS researchers also gave a number of talks with accompanying demonstrations in exhibit booths, including:

- Chuck Koelbel (Rice), Rich Wolski (UCSB), and Dan Reed (then still UNC) gave overviews of the VGrADS project in the GCAS booth (a collaboration of Rice, UH, and Texas A&M), SDSC booth, and RENCI booth.
- Lavanya Ramakrishnan (IU), Anirban Mandal (UNC), and Dan Nurmi (UCSB) led a team of students and staff in demonstrating the LEAD application running with VGrADS for both performance and fault tolerance.
- Several members of the VGrADS team, including Chuck Koelbel, Kiran Thyagaraja (Rice), and Mark Huang (UH) were also active in Grid-related talks and demos for the TIGRE project.
- Dan Reed and Lavanya Ramakrishnan also presented a VGrADS-related poster "Performability Modeling for Scheduling and Fault Tolerance Strategies for Grid Workflows".

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

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 have continued development of the Virtual Grid Execution System (vgES) for managing the abstractions that are key to our work. Key contributions in past years included the introduction of slotted virtual grids, which allow unified management of reserved resources and resources controlled by batch queues and resource equivalence, by which an application can identify trade-offs between processor architectures.

  In this reporting period, VGrADS researchers have investigated the degree to which fault-tolerance can be effectively implemented "underneath" virtualized execution abstraction (e.g. the VGrADS slot). This capability addresses a long-standing question over the degree to which grid software technologies can hide implementation complexity from application developers. It also extends the contributions described in previous reports in unifying the virtual grid treatment of fault tolerance.

- VGrADS researchers developed a temporal reasoning framework to support performance validation and diagnosis of long-running grid applications for virtual grids. In particular, this reasoning framework supports monitoring of grid applications, allowing them to identify changes in conditions that require adaptation. The framework's qualitative performance analysis can help bind expectations of grid applications with resource behavior in the Virtual Grid Execution System.

- VGrADS researchers developed fault-tolerance and recovery algorithms for reliable execution of scientific workflows on computational grids and validated them using meteorological workflows from LEAD. In particular, the fault-tolerance techniques increase the reliability of workflow executions through over-provisioning and migration of workflow steps.

- VGrADS researchers have continued development and evaluation of grid scheduling heuristics. We previously presented a two-phase strategy of a simple virtual grid selection phase (picking "good" resources to run on) followed by good scheduling heuristics (such as the HEFT algorithm) for optimizing workflow application performance. We also developed schedulers that combine

estimations of application performance and batch queue wait times to generate high-quality schedules in slotted virtual grid environments. This year, we extended those static methods to a hybrid dynamic/static scheduler that revises its choices based on run-time feedback. We found that, on a real grid with varying load, such a method not only improved on purely static methods (as one would expect), but also was better than a purely dynamic method.

- VGrADS researchers have investigated the feasibility of implementing cloud-computing services in research and scientific computing contexts. Commercial cloud-computing services are well suited to web-service deployment and large-scale text search. Eucalyptus demonstrates that these services can be provided by existing, locally deployed clusters that are servicing a scientific user community.

- VGrADS researchers have demonstrated that effective batch-scheduling protocols with provable incentive properties (e.g. incentive compatibility) can be developed, both from a theoretical perspective and in implementation.

- VGrADS researchers have studied the feasibility of traditional compiler optimizations for grid computing systems. Previous reports have documented the effectiveness of optimizations such as register allocation in the context of just-in-time compilation. This year, we have studied the requirements of system-specific auto-tuning of FFT software, which can be used to significantly improve individual node performance in a grid environment. 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, with excellent performance (comparable to MPICH2 or LAM). 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. New work this year has included evaluation of the Binomial Graph Network as a basis for fault-tolerant message passing layers.

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. Notably, this includes discussion and collaboration with several world-renowned researchers from other institutions with whom the students would not normally interact.

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. In the Spring 2008 semester, Lennart Johnsson taught *COSC 6365: Introduction to High Performance Computing* at UH. In the Spring 2008 semester, Jack Dongarra taught *CS 594-004: Scientific Computing for Engineers* at UTK.

VGrADS researchers and staff have been actively involved in efforts to encourage middle-school and 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 middle- 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 SC'07 conference in Reno, Nevada (November 10–16, 2007). VGrADS activities at SC'07 are discussed under both "Outreach Activities" and "Project Activities."

4) Contributions to Resources for Research and Education

There is nothing to report at this time.

5) Contributions Beyond Science and Engineering

There is nothing to report at this time.