I. Project Activities

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

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

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

VGrADS is pursuing this agenda by collaborating with leading scientific applications to elicit key challenges, validate results, and disseminate technology. It will distribute software that it creates in open-source form for the research community. It 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 (6/1/06 – 5/15/07), 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 changed significantly during the funding period due to the death of VGrADS PI Ken Kennedy in February 2007. However, all other active PIs have remained with the project, and the overall technical direction has not changed as a result of this tragedy. Details of the new management structure are discussed in *Management & Structure 4)*.

Finally, the VGrADS milestones have been revamped to reflect both management and personnel changes during the funding period and plans for the final year of the project. Annotated VGrADS milestones appear in *Project Milestones (Section 5)*.

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

VGrADS research has always been driven by the needs of actual applications. Initially, we selected four applications (EMAN, EOL, GridSAT, and LEAD) based on our experience in the GrADS project and from other sources to help derive requirements for Virtual Grid (VG) functionality and to serve as tests of new tools methods. We have subsequently completed our study of EOL, but have added an effort on 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

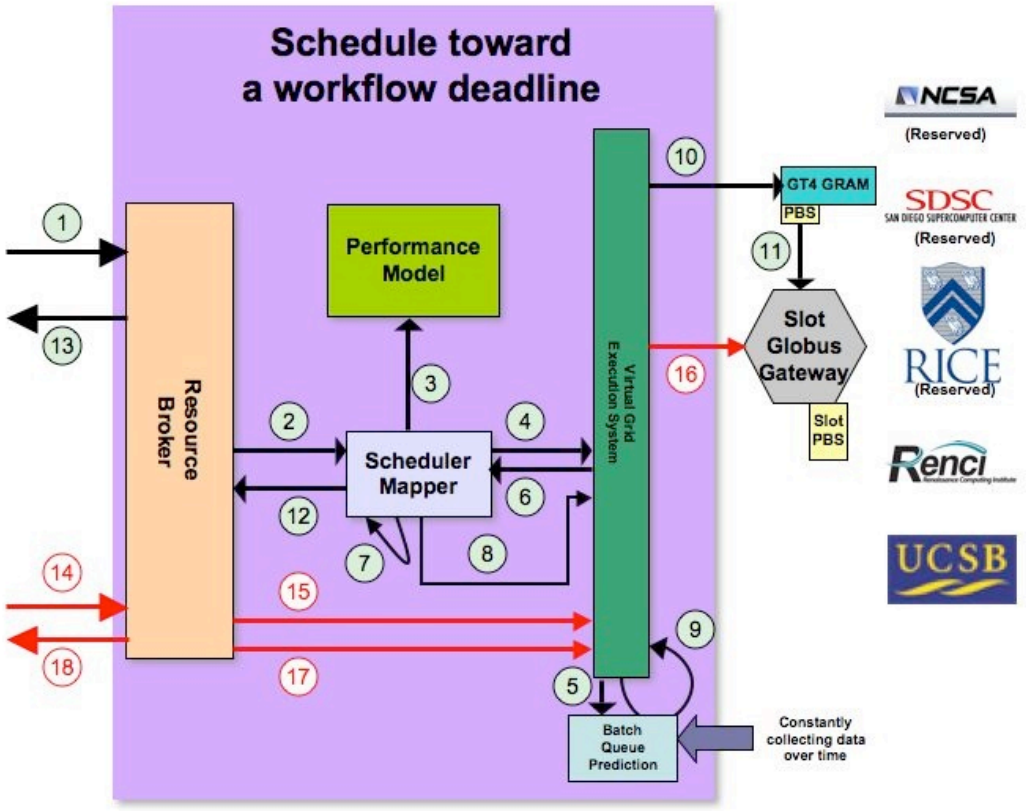
VGrADS is closely collaborating with another NSF funded ITR project, LEAD (Linked Environments for Atmospheric Discovery). The goal of LEAD is to build a scalable web services workflow infrastructure for meteorological data and models. The meteorology and the web services components of LEAD are being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

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

We 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. She has 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 persistent 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 completed the first VGrADS/LEAD integration and demonstrated it at SC'06. This work was a true collaboration, involving Lavanya Ramakrishnan (originally UNC, later moving to Indiana), Yang-Suk Kee (ISI), Dan Nurmi (UCSB), Ryan Zhang (Rice), and many others. Among other benefits, this exercise introduced and refined the resource slot concept, showed how the Batch Queue Prediction mechanism could be used to manage slots, and demonstrated slot-based application scheduling based on performance models. The net effect was that LEAD could produce an abstract workflow (as a DAG with constraints) and have the workflow scheduled and executed on VGrADS resources. The figure below shows some of the steps required to accomplish these tasks.



Black arrows correspond to the mapping and scheduling steps:

1. LEAD Workflow Configuration Service (invoked from LEAD portal) passes DAG and constraints (e.g. deadlines) to VGrADS-based Resource Broker.
2. Resource Broker passes DAG, constraints, and a pointer to the application performance model to the VGrADS mapper.
3. VGrADS mapper queries the performance model for the task's resource requirements to meet the constraints.
4. Mapper requests slots from the Virtual Grid Execution System (vgES) using `vgFind` command.
5. vgES queries the Batch Queue Prediction (BQP) 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 BQP data, vgES chooses slots and returns a representation to the mapper.
7. Scheduler assigns DAG nodes to slots, using the performance model to predict computation and communication time. Return with an error if no schedule can be found; otherwise, continue to next step.
8. Scheduler obtains resources by sending `vgBind` command to vgES.

9. For batch-queued resources, vgES queries BQP (repeatedly) for time to submit job.
10. 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.
11. Each slot is given a PBS queue, accessible via a Globus gateway.
12. When all slots are bound, the mapper returns its mapping to the resource broker.
13. The resource broker annotates the DAG with the resources and returns to the LEAD portal.

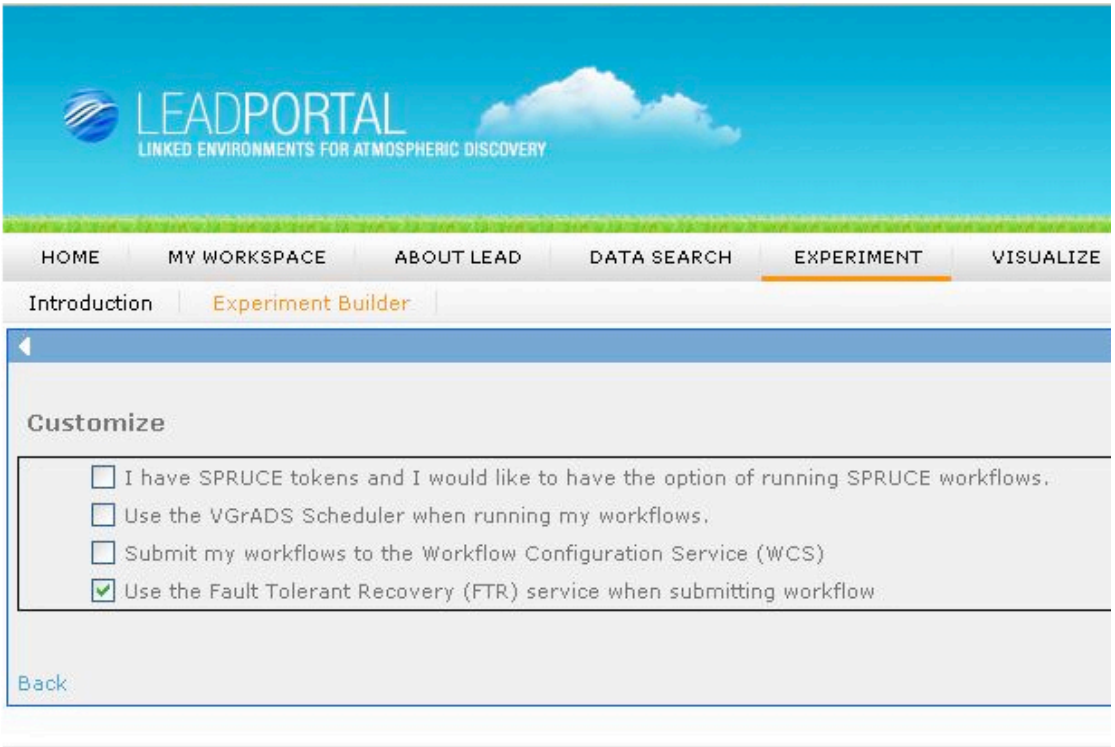
Red arrows correspond to the later steps in executing the workflow:

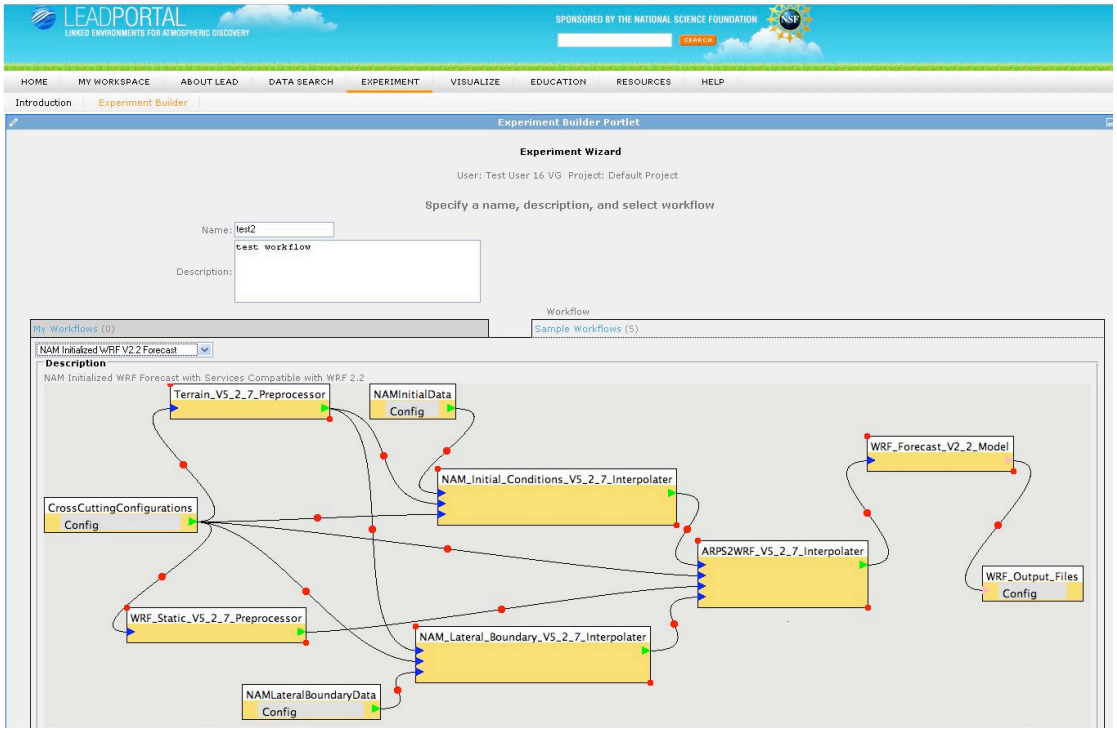
14. The LEAD workflow engine, working through an application service, invokes a task (DAG node) to run.
15. The resource broker sends a `vgLaunch` command for the task to vgES.
16. vgES runs the task on the allocated resource gateway.
17. While the job executes, the resource broker issues `vgStatus` commands to monitor progress.
18. On job completion, the resource broker notifies the LEAD application service and workflow engine.

Since the conference, Anirban Mandal and Gopi Kandaswamy (both UNC) have extended the LEAD scheduler with a prototype Fault Tolerance/Recovery Service (FTR). FTR attempts to choose the best fault tolerance mechanism for individual components using simple reliability models. Given a reliability model for the component, a deadline *d*, and a required probability of success *x*, FTR considers:

- Restart: Find the resource that best meets the *d* and *x* constraints. Run the component there. If the resource fails during execution, retry the component.
- Overprovisioning: Find *P* resources such that the probability of all failing is at most *x* and all resources will complete by *d*. Start the component on all resources.
- Migration: Find a sequence of resources $\{s_1, s_2, \ldots, s_P\}$ such that the probability of all failing is at most *x*, and the sum of their expected up times plus migration costs between them is less than *d*. Start the component on *s1*, with checkpointing enabled. If *s1* fails, migrate the checkpoint to *s2* and restart there. Continue until the computation completes.

The UNC researchers demonstrated the overprovisioning and restart capabilities at the VGrADS internal workshop in April 2007. Although it was still in the very early stages, these screenshots suggest the possibilities ahead.

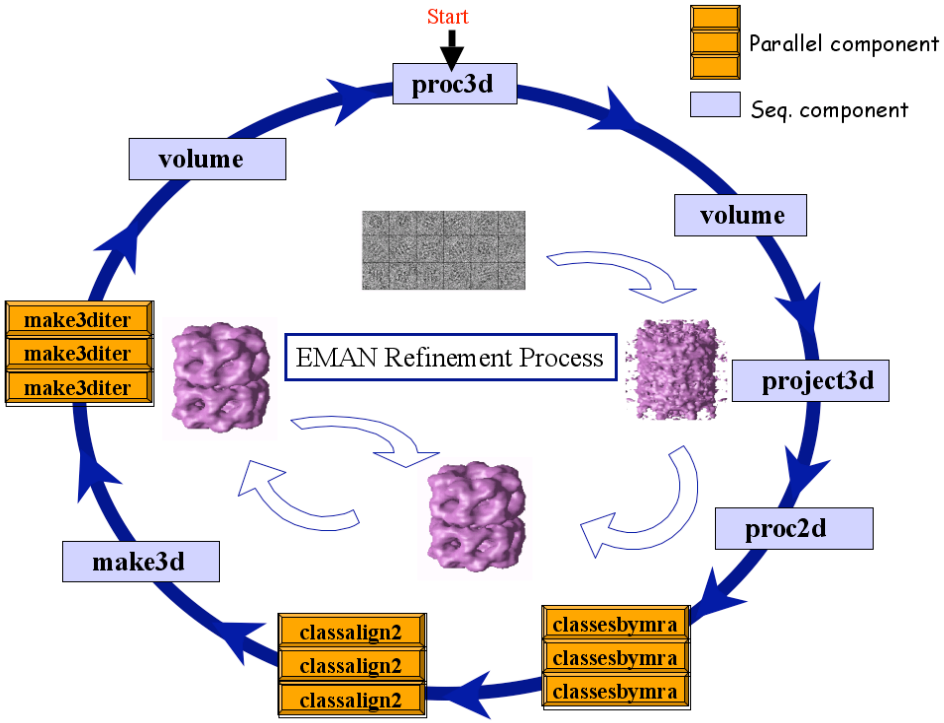

LEADPORTAL
LINKED ENVIRONMENTS FOR ATMOSPHERIC DISCOVERY
HOME
MY WORKSPACE
ABOUT LEAD
DATA SEARCH
EXPERIMENT
VISUALIZE
Introduction
Experiment Builder
Customize
I have SPRUCE tokens and I would like to have the option of running SPRUCE workflows.
Use the VGrADS Scheduler when running my workflows.
Submit my workflows to the Workflow Configuration Service (WCS)
Use the Fault Tolerant Recovery (FTR) service when submitting workflow
Back


LEADPORTAL
LINKED ENVIRONMENTS FOR ATMOSPHERIC DISCOVERY
SPONSORED BY THE NATIONAL SCIENCE FOUNDATION
HOME
MY WORKSPACE
ABOUT LEAD
DATA SEARCH
EXPERIMENT
VISUALIZE
EDUCATION
RESOURCES
HELP
Introduction
Experiment Builder
Experiment Builder Portlet
Experiment Wizard
User: Test User 16 VG Project: Default Project
Specify a name, description, and select workflow
Name: test2
Description: test workflow
Workflow
My Workflows (0)
Sample Workflows (5)
NAM Initialized WRF V2.2 Forecast
Description
NAM Initialized WRF Forecast with Services Compatible with WRF 2.2
Terrain_V5_2_7_Preprocessor
NAMInitialData Config
WRF_Forecast_V2_2_Model
NAM_Initial_Conditions_V5_2_7_Interpolater
CrossCuttingConfigurations Config
ARPS2WRF_V5_2_7_Interpolater
WRF_Output_Files Config
WRF_Static_V5_2_7_Preprocessor
NAM_Lateral_Boundary_V5_2_7_Interpolater
NAMLateralBoundaryData Config

Plans going forward include enhancements to vgES and NWS to provide additional reliability data, extensions to the VGrADS and LEAD workflow execution architecture to allow better rescheduling, and improvements to the FTR decision mechanisms for component- and workflow-level fault tolerance.

1.2 EMAN

Much of the early VGrADS application research efforts 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.

In this reporting period, the primary development regarding EMAN was its use 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. For clusters with a batch queue, the time for a job in queue can be highly variable and potentially long – longer in many cases than the execution time itself. Hence, considering queue wait times while scheduling is absolutely essential. So, we equipped the scheduler to use predictions of when resources become available (using novel NWS methodologies developed at UCSB) along with accurate performance models to decrease the overall turnaround time of the workflow when executed on non-dedicated systems. This technology was first demonstrated at SC'05, and published as a paper at SC'06. The results showed that using batch-queue wait time predictions significantly reduces the turnaround time of EMAN computations.

We have long planned to tackle a "challenge" problem with our Grid-enabled EMAN. Our collaborators at BCM have even larger data sets, and the State of Texas has funded a large Grid activity (the TIGRE project, http://www.hipcat.net/Projects/tigre, and the LEARN network, http://www.tx-learn.org/). Preliminary results of the new data are already in press, but even small amounts of additional resolution (derived from additional iterative processes on extended data sets) would enable important new biological findings. We believe that the combination of VGrADS scheduling and the resources available through TIGRE will allow us to achieve additional resolution. However, this development is currently not being pursued aggressively, as TIGRE has

not yet finalized agreements for sharing computational cycles. (Also, our applications efforts have shifted to LEAD as noted in section 1.1.)

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) which results in the expression evaluating to TRUE?' The system stands apart from other SAT solvers in the sense that it was designed explicitly to run in grid environments, and has built-in intelligent parallelism scheduling mechanisms. As a result of this design, the system has been used successfully and quickly to solve several previously unknown problems by utilizing vast amounts of computational resources.

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

As a VGrADS driving application, GridSAT motivates both the functionality and the performance of the virtualized resource discovery and allocation mechanisms. The GridSAT scheduler considers resources abstractly, strictly in terms of their performance characteristics. VGs 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. We are currently considering a port to vgES, as well as other updates and modifications.

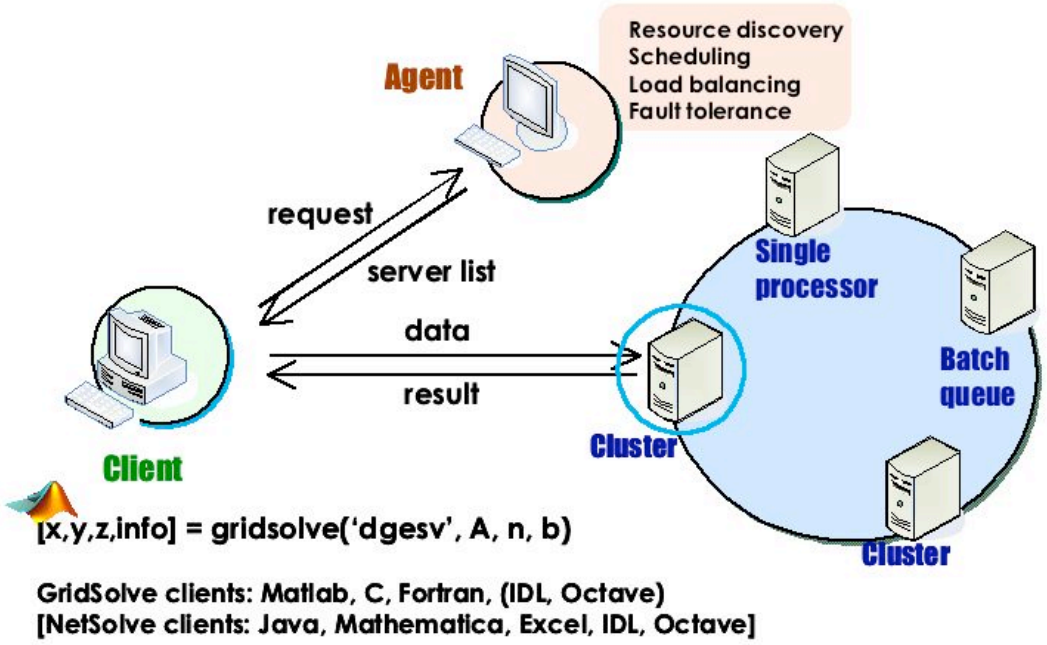
1.4 FT-LA: Fault Tolerant Linear Algebra

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. Currently, GridSolve uses a static scheduling model based on performance prediction from historical data; moving to VGrADS would allow GridSolve (in principle) to take advantage of the batch queue scheduling method. Moving forward, GridSolve could also take advantage of fault tolerance capabilities as they are added to vgES. We are exploring these options for work in the next year.

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 seven research thrusts this year:

1. Improved scheduling for workflow computations on VGs,
2. Scheduling workflow computations on batch queue controlled resources,
3. Scheduling applications for reliability as well as execution time,
4. Using economic methods for resource allocation,
5. Performance prediction of application components to be mapped onto the Grid,
6. Compiling and optimizing node programs for use in a Grid environment, and
7. Fault-tolerant libraries for MPI and OpenMP.

The following subsections discuss each thrust in turn.

2.1 Scheduling Applications on Virtual Grids

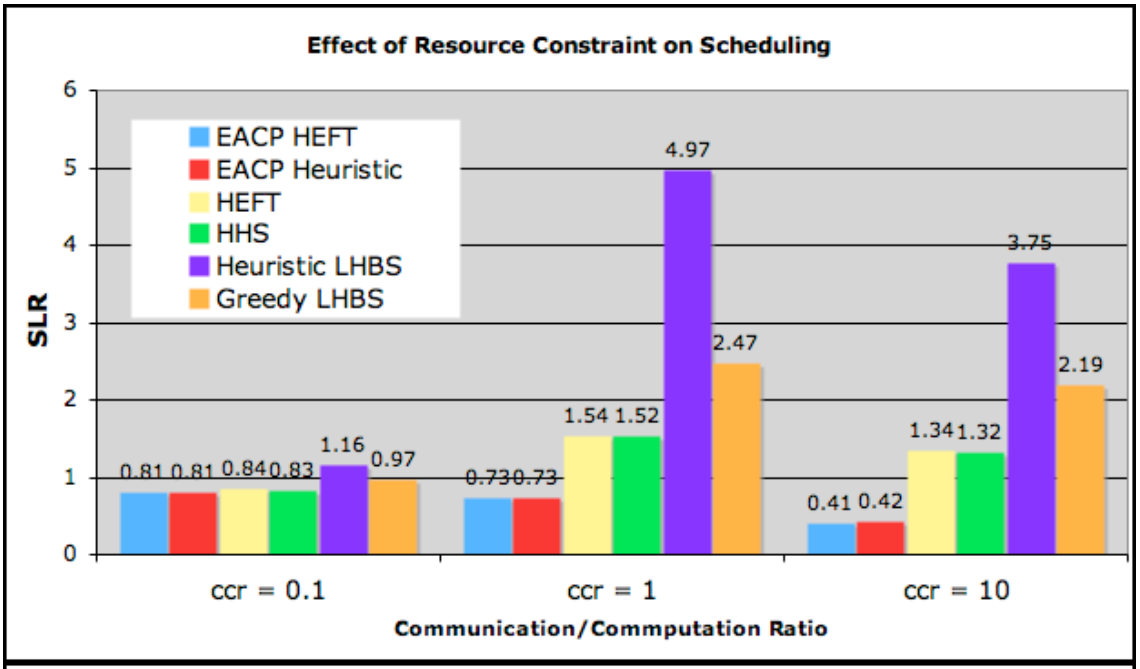
As we have previously reported, researchers at Rice and UCSD have developed scheduling strategies for EMAN, EOL, and other workflow applications. These applications are modeled by directed acyclic graphs (DAGs), where the nodes are computations (which may themselves be parallel structures) and the edges 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.

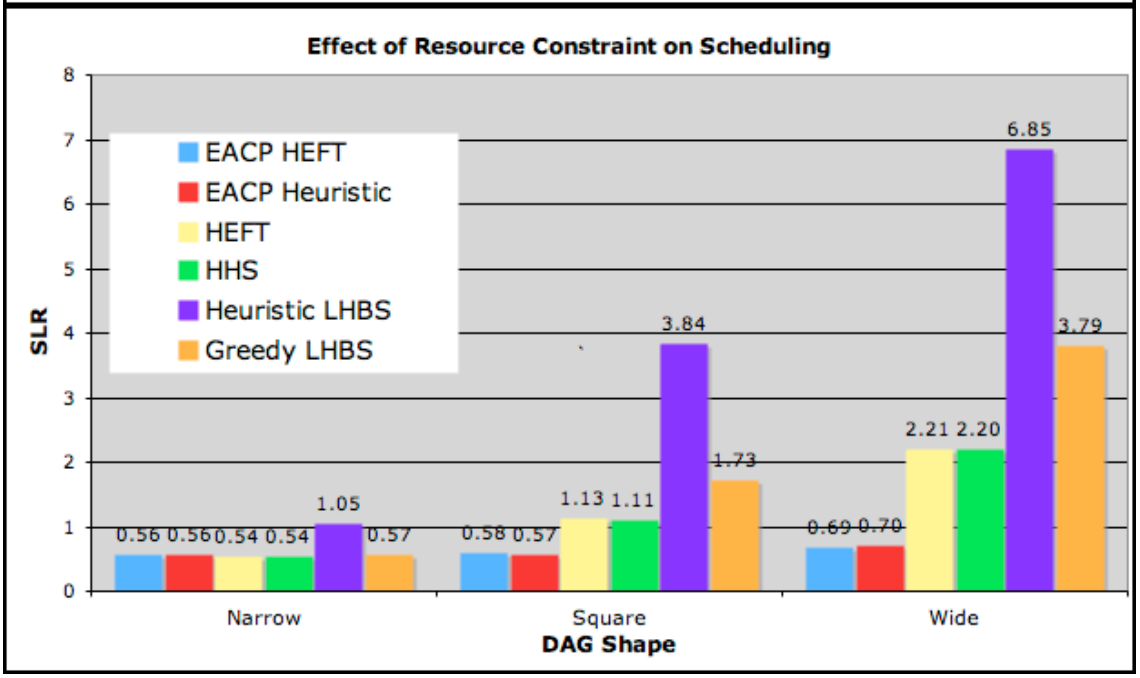
One excellent example of this methodology was the paper "Performance of Scheduling Algorithms in Grid Environments" by Ryan Zhang and VGrADS PIs Charles Koelbel and Ken Kennedy (all from Rice University) at the CCGrid'07 conference[1]. Zhang performed a large-scale simulation study of the makespan (computing plus communication time) to process 10,000 DAGs using several schedulers. Among the conclusions were that

- List-based scheduling methods (such as Heterogeneous Earliest Finish Time (HEFT)) on average outperform level-based methods (such as Levelized Heuristic Based Scheduling (LHBS), developed earlier in the VGrADS project).
- The critical scheduling decision on grids is the choice of resources to use (i.e. the choice of VG).
- Estimated Aggregate Computing Power (EACP) is a viable means of selecting clusters to use in a grid computation.

As examples, we reproduce here two charts from the paper showing the sensitivity of various DAGs to the scheduling algorithm. To summarize them, EACP is particularly effective for DAGs with high communication to computation ratios (CCR) and high levels of parallelism (wide DAGs). The vertical axis – Schedule Length Ratio (SLR) - is a normalized performance measure.

[1] The paper has been nominated for the Best Paper award at the conference. At this writing, the conference has not yet taken place, but we will report if it is honored.

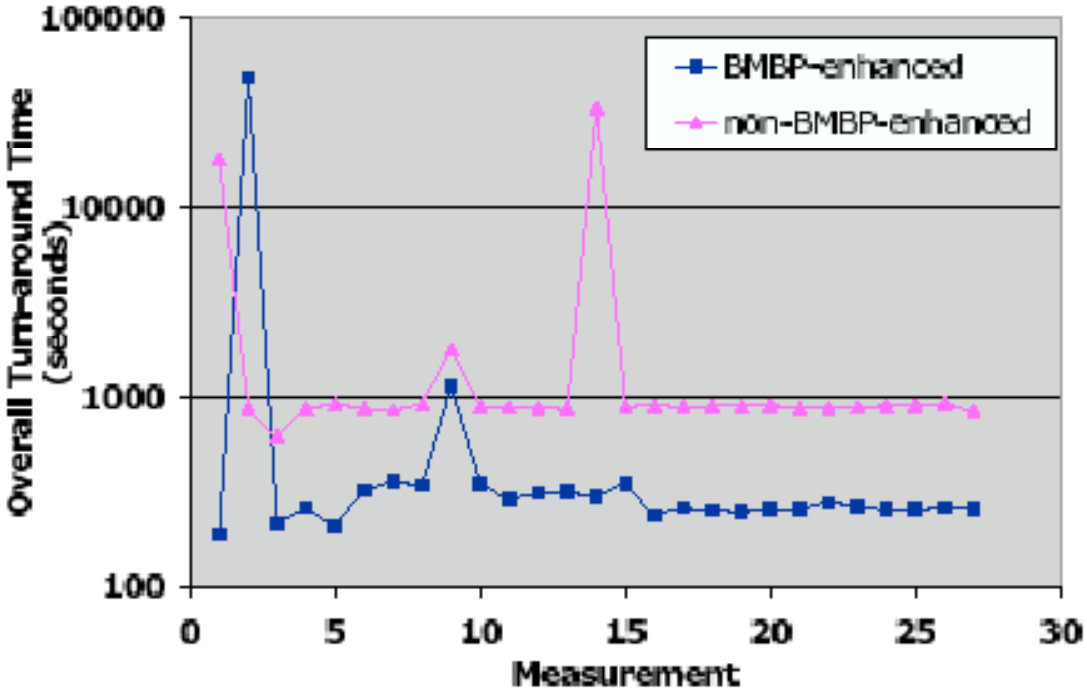




2.2 Scheduling Applications on Batch Queues

Researchers at UCSB and Rice have collaborated on a prototype scheduler for Grids in which resources are controlled by batch queues. We reported on the first version of this research last year, when it formed the basis for our SC'05 demonstrations. This year, we

analyzed the data gathered in that development and reported it in "Evaluation of a Workflow Scheduler Using Integrated Performance Modeling", a paper by Nurmi et al at SC'06. One of the key graphs is reproduced below, showing that scheduling that takes batch queue wait times into account ("BMBP-enhanced") improves total turnaround time by a factor of three on most tests.



These results spurred us on to include batch queue prediction as a key mechanism in implementing slotted virtual grids, which we demonstrated for the LEAD application at SC'06.

2.3 Scheduling for Reliability

Researchers at UTK are also extending scheduling concepts to deal explicitly with dynamic environments. Most workflow scheduling algorithms are based on the *deterministic* model, driven by (assumed) accurate knowledge of the tasks and their performance. Although a schedule may be optimized with respect to these assumptions, it may perform poorly in real systems owing to run-time variations. In particular, inaccurate assumptions that the resources will stay up for the duration of the application are likely to make the overall schedule highly unreliable.

To address this problem, VGrADS PI Jack Dongarra and co-workers have designed scheduling algorithms to simultaneously minimize makespan and maximize reliability. Although these objectives are often in conflict, it is possible to define (at least theoretically) a Pareto trade-off curve between them. An application can then choose an appropriate level of reliability (e.g. 90% chance of completing the application) and minimize its time with this constraint. Equally, it could choose a target time (e.g. a deadline to meet) and find the schedule most likely to complete. Based on the provable fact that choosing processors with minimal {execution time}×{failure rate} maximizes reliability, the UTK team suggests ordering the Grid processors by this quantity and scheduling on the minimal number of processors that meets the deadline. Moreover, they show how this metric can be incorporated into the HEFT scheduler, providing a way to schedule for maximum reliability. The details of this process are incorporated in the paper "Bi-objective Scheduling Algorithms for Optimizing Makespan and Reliability on Heterogeneous Systems" by Dongarra et al. at SPAA'07.

2.4 Resource Allocation via Grid Economies

Researchers at Rice and 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 (like those described in section 2.5) 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 had explored in past work.

The current embodiment of this idea is a simulator that takes workflow DAGs, with cost models for each node, and computes an instantaneous equilibrium price for each resource. Each application has a budget that it can divide among its tasks as it sees fit. The simulator operates by the tâtonnement mechanism (invented by Walras in 1874) in which a market maker (e.g. a resource manager like vgES) posts prices for all resources, and all applications respond with offers to buy resources. The market maker then increases prices on over-subscribed resources, cuts prices on under-subscribed resources, and requests updated bids. This continues until prices converge and supply equals demand. In our case, each application bids for the cluster that is most cost-effective based on its performance model and the posted prices.

Under certain conditions on the market, this is guaranteed to find an equilibrium price that is economically efficient. We are internally debating whether these conditions are in fact met in our experimental environment. Some problematic conditions for convergence include "herd behavior" (when all applications flock to the same resources) and price granularity (when applications suddenly increase their demand at certain price thresholds). We have now found settings for the tâtonnement auction that seem to converge, but still have many tests to run.

Once we have determined the feasibility of grid economies, we hope to use the method as a means of selecting VGs and as the basis for a new scheduling algorithm. This work is, however, very preliminary.

2.5 Automatic Construction of Performance Models

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

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

Although this research has been very valuable, we have devoted more efforts recently to empirical models based on archived performance information. Simple polynomial extrapolation of performance databases is commonly used in many applications (including LEAD) as a quick-and-dirty model, and this seems to be accurate enough for our purposes.

2.6 Compiling and Optimizing Node Programs

Researchers at Rice have pursued methods for compilation on individual Grid nodes under the aegis of VGrADS programming tools. As detailed in previous reports, we have adopted a strategy of using LLVM (from Adve's group in Illinois) as a platform to enable compilation on heterogeneous processors. In effect, we perform Just-In-Time (JIT) compilation of the application when we assign it to a Grid node. The most notable result of this work in this reporting period was the graduation of Anshu Dasgupta, a PhD student of VGrADS PI Keith Cooper, who developed versions of important compiler optimizations applicable to a JIT context. The key technical constraint for

these new versions was low complexity for the compiler method, since it was invoked at application run time.

2.7 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. FT-MPI is built upon the fault tolerant HARNESS runtime system and framework for distributed computing that is also being developed at UTK. FT-MPI survives the crash of n-1 processes in an n-process job and, if required, can restart them. However, it is still the responsibility of the application to recover the data-structures and the data on the crashed processes.

FT-MPI 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. A completely new MPI-2 compliant implementation, OpenMPI will offer advantages for system and software vendors, application developers, and computer science researchers.

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

The latest research directions address the continuing development of vgES (Virtual Grid Execution System) to support complex, adaptive workflow applications. We are currently focused on three distinct areas. First, we have refined the support for including resource availability time in a vgDL request. The previous version of the Virtual Grid had a crude method to allow vgDL requests to specify *when* resources should be made available; that version dealt exclusively with time-sharing or dedicated resources that were made immediately available to the user. Our new version refines the time notion in the vgDL language and accommodates batch queues. The key motivating application for this work is the LEAD workflow. Its requirements are indicative of the complicated machinations required for adaptive, real-time applications. Second, we have enhanced the language's ability to successfully find the resource set that matches the application's resource request. We do this by expressing resource equivalence. Finally, we redesigned our execution system as a generic framework for resource instantiation and execution environment, developing a plug-in architecture to a variety of resource managers.

In parallel with the restructuring of the virtual grid execution system, we conducted research in the five 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), temporal reasoning framework for performance validation (Section 3.7), and Grid resource specification generation (Section 3.8).

We demonstrated the second-generation vgES as a framework for resource management and an execution environment, including the slot management, at the SC'06 Conference in Tampa, FL. This demonstration illustrated the power and flexibility of vgES by building an application manager for a Grid-enabled application in production, LEAD (see section 1.1). Based on performance models, the application manager allocated multiple resource collections across TeraGrid and private institute resources to satisfy different slots using both advance reservations and virtual reservations realized by the batch queue prediction tool. 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.

3.1 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.

3.2 Slot Allocation over Advance Reservation and Virtual Reservation

A slot is a high level representation of resources in time and space. A slot 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 the Virtual Grid is a resource collection in time and space across multiple resources from the application viewpoint. The user can specify time constraints on aggregates, using the following vgDL slot construct.

slot := "<" start "," duration ">"

We implemented this slot concept against advance reservation and virtual reservation using batch queue prediction. 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).

3.3 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.

3.4 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, during the reporting period 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 at SC'06. In addition, we have authored a paper analyzing VARQ's effectiveness for submission to SOSP 2007 (currently under consideration). QBETS research, which has produced several published works (PPoPP06, IISWC06), 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 Temporal Reasoning Framework for Performance Validation

We have also continued the development of a qualitative temporal reasoning framework to support performance validation for VGs. Our goal is to reason about temporal events to differentiate between severe, persistent behavioral violations and transient ones. This will help bind the expectations of applications with the resource behavior in the VG execution system. The temporal reasoning framework supports grid environments to (a) monitor and validate the behavior of long-running scientific workloads, (b) diagnose possible sources of behavior changes, and (c) provide reasoning support for applications to adapt to variations observed.

The temporal reasoning framework gathers performance traces from multiple monitoring levels. The framework generates qualitative temporal signatures from these real-time, multi-level performance data and compares these signatures to expected behavior. In case of any mismatch, the framework hints at the causes of altered performance and suggests potential solutions to an application user or system administrator.

In the reporting period, we have focused on expanding experimental validation, refining algorithms, and prototyping the framework.

Experiments with grid workflows from meteorology (WRF) and astronomy (Montage) reveal common qualitative temporal signatures characterizing successful and well-performing execution. This allows the system to learn and store the signatures of typical temporal behaviors. We performed these experiments on four different architectural configurations, typical of existing grid resources, including the NCSA IA-64 TeraGrid cluster and RENCI's BlueGene/L.

We have also refined the algorithms used for time series pattern recognition by 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 better visualization of the multi-dimensional data contained in the behavioral signatures generated. Among the tools available, we have considered parallel coordinates, Andrew's plots, and star glyphs as best informative choices.

Furthermore, we are currently in the process of completing experiments with stimuli affecting performance; most result in signatures with distinct characteristics. The ability to distinguish between qualitative signatures of expected states supports the performance validation and diagnosis of long-running scientific grid applications.

3.8 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 (in IPDPS 2006) 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 has been accepted for HPDC 2007. We have submitted the validation results to SC 2007 (currently being reviewed).

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. Project design and coordination during the current reporting period were enabled through weekly technical teleconferences involving researchers from each of the VGrADS sites, two PI meetings, one VGrADS executive committee meeting, VGrADS planning workshops at UCSB (9/7-8/06) and Rice (4/18-19/07), one developer's workshop at USC/ISI (8/9-11/06), and communication via VGrADS mailing lists. The fall 2007 VGrADS planning workshop will be held at UNC (10/11-12/07). 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 February 7, 2007, VGrADS PI Ken Kennedy died after a long battle with cancer. (See http://www.media.rice.edu/media/NewsBot.asp?MODE=VIEW&ID=9268 for details.) Following Ken Kennedy's death, VGrADS co-PI Keith Cooper at Rice University assumed the role of lead VGrADS PI. Rice University remains the lead VGrADS institution. Partners participating in the VGrADS project prior to Ken Kennedy's death continue to contribute to the ongoing success of the project. Keith Cooper 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)
Dan Reed (UNC)
Rich Wolski (UCSB)

The executive committee will meet on a regular basis to review progress and milestones, discuss plans for the future, and advise the PI on resource allocation issues. The VGrADS executive committee met at Rice on April 19, 2007 and plans to meet at UNC on October 11, 2007. In addition, the VGrADS executive committee plans to meet as needed by teleconference.

4.2 VGrADS Web Site

During the funding period, the VGrADS Web site, http://vgrads.rice.edu, was restructured and rewritten 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 examined and modestly revised the original research milestones for the project. We first reported the revised milestones in our 2006 annual report. In this section, we report on progress toward the new milestones.

5.1 Year 3 Milestones

We report on Year 3 milestones here because work on some of them was in progress at the time of the last annual report.

Our milestones are

i. Execution System/Virtualization:
   - Investigate novel vgES resource selection and binding techniques and evaluate via simulation of large-scale environments
     (Done: vgES initial implementation and evaluation described in 2006 annual report)
   - Develop and evaluate techniques for scheduling in VG environments, exploiting synergy between resource specification and scheduling
     (Done: papers described in 2006 annual report; see also sections 2.1 and 3.8 above)
   - Augment vgES to include time-dependent descriptions via probabilistic and reservation-based provisioning
     (Done: slotted virtual grids described in 3.1 above; reservation-based and probabilistic slot management described in 3.2 and demonstrated at SC'06 conference)
   - Conduct experiments with the time-space reasoning contracts in virtual grid environments
     (Done: see section 3.7)

ii. Execution System/Grid Economy:
- Begin designing experiments to test pricing techniques using VGrADS framework.
  (Done: initial tests described in 2006 annual report, continuing work described in section 2.4 here and in SC'06 paper by Nurmi)
- Enhance availability and batch-queue predictions to be suitable for scheduling, particularly with respect to virtual reservations.
  (Done: described in 2006 annual report)
- Continue simulation experiments to evaluate resource allocation efficiency.
  (Done: described in 2006 annual report and in HPDC papers)

iii. Execution System/Fault Tolerance:
- Consider novel techniques and integrate predictive tools
  (Done: described in 2006 annual report)
- Develop a multi-level fault tolerance API to capture and adapt to failures at different levels and integrate with the virtual grid
  (Done: described in 2006 annual report)

iv. Programming Tools/Workflow:
- Evaluate and refine multi-level scheduler(s) for workflow applications
  (Done: described in 2006 annual report; also reported in CCGrid'07 paper)
- Adapt workflow scheduling to take advantage of new time-dependent descriptions available in vgES
  (Done: used in LEAD/VGrADS demonstration at SC'06)
- Explore novel workflow schedulers incorporating new resource behaviors (e.g. batch queue delays)
  (Done: described in 2006 annual report; also included in SC'06 paper)

v. Applications:
- Produce a revised implementation EMAN with queue-based scheduling enabled
  (Done: described in 2006 annual report)
- Collaborate with LEAD application team on providing new capabilities for resource scheduling
  (Done: described in 2006 annual report)
- Evaluate fault-tolerance and virtual grid APIs with LEAD application
  (Done: described in 2006 annual report)

vi. Education, Outreach, and Training:
- Continue AGEP program
  (Done: three AGEP students supported in 2006)
- Participate in Grace Hopper Conference
  (Done: 13 students and one staff supported to Grace Hopper 2006)
- Continue program of grad student exchange and collaboration
  (Done: collaboration is widespread)

5.2 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: demonstrated at SC'06; see also sections 3.1 and 3.8)
   - Explore techniques to extend the VG abstraction to diverse resource management paradigms (e.g., probabilistic space-time resource abstractions).
     (Done/in progress: part of slotted VG work described in section 3.1)
   - Expand techniques for integrated time-space reasoning with performance/fault tolerance capabilities
     (In progress: see work described in section 3.7)

ii. Execution System/Grid Economy:
   - Complete integration of new prediction capabilities with vgES to support both scheduling and grid economy work.
     (Done/in progress: scheduling via predictions described in SC'06 paper and used in SC'06 demonstration (see sections 1.1 and 3.4); grid economy work described in section 2.4)
   - Conduct empirical investigation of pricing scheme and its effect on resource allocations.
     (In progress: ongoing work described in section 2.4)

iii. Execution System/Fault Tolerance:
   - Implement novel techniques (e.g. diskless checkpointing in a general setting in Open-MPI)
     (Done: OpenMPI work described in 2006 Euro PVM/MPI paper and section 2.7)
   - Prototype and experiment with techniques to implement dynamic adaptation in a multi-level fault tolerance environment on the virtual grid
     (Done/in progress: FTR prototype is running (see section 1.1), but extensions are still under development)

iv. Programming Tools/Workflow:
   - Demonstrate novel, scalable workflow scheduler(s) on TIGRE grid
     (In progress: TIGRE applications needed for experimentation)
   - Explore robustness of workflow schedulers, particularly with regard to fault tolerance
     (Done: results reported in CCGrid'07 and SPAA'07)

v. Applications:
   - Incorporate novel, scalable workflow scheduler(s) into EMAN and LEAD applications
     (Done: used in LEAD/VGrADS demonstration at SC'06)

- Continue evaluation of fault tolerant techniques with LEAD
  (Done: part of FTR prototype described in section 1.1)

vi. Education, Outreach, and Training:
- Continue AGEP program
  (In progress: will support at least two students in summer 2007)
- Participate in Tapia Symposium
  (In progress: VGrADS co-PI is poster chair, will support student travel)
- Continue graduate student exchanges and collaborative workshops
  (Done: collaboration continues)
- Participate in Grace Hopper Conference
  (Moved milestone from Year 5 due to rescheduling of Grace Hopper conference)
  (In progress)

5.3 Year 5 Milestones

We have also made some progress on a few Year 5 milestones, and updated some others. 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
  (This represents a clearer focus than our previous milestone, "Improve resource selection and binding techniques for VGs based on insights from Year 4".)
  (The work described in section 3.8 is a first step in this direction, as are the HPDC papers in the bibliography.)
- Improve resource-scheduling techniques for VGs that consider resource efficiency and cost
  (This is also a more focused version of a previous milestone.)
  (The work described in sections 2.3 and 2.4 is a step in this direction.)
- Prototype and evaluate extended VG abstraction over environments with diverse resource management paradigms.
  (The LEAD/VGrADS demonstration at SC'06 is a prototype of this management, which we are further developing and evaluating.)
- Demonstrate vgES with resource selection, resource binding, VG scheduling, for application kernels across large-scale grid platforms with diverse resource management paradigms.
  (The LEAD/VGrADS demonstration at SC'06 arguably meets the letter of this milestone, but we are continuing development.)
- Validate and assess the integrated resource provisioning policies
  (The work described in section 3.8 will be part of this assessment.)

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.
  (This is a new milestone building on the work described in section 2.4.)
- Design experiment to investigate allocation efficiency under various pricing schemes.
  (Done. We have investigated both Smale's method and tâtonnement methods in this context.)
- Target second VGrADS-enabled application (to be determined) as a driving application.
  (The work described in section 2.4 does this.)
- Verify using both GridSAT and second application.

iii. Execution System/Fault Tolerance:
- Integrate fault tolerance features into vgES and test on LEAD application
  (This replaces the less specific and less ambitious milestone "Limited validation and assessment".)
  (This extends developments described in sections 1.1, 1.4, and 3.6 and the Year 4 milestones.)

iv. Programming Tools/Workflow:
- Incorporate novel techniques from vgES and fault tolerance work into workflow schedulers
  (Sections 2.3 and 3.6 report preliminary work in this area.)
- Study new problems in workflow based on grid economies, fault tolerance, and dynamic resource behavior
  (This focuses a less specific previous milestone.)
- Consider compilation approaches to task migration for fault tolerance
  (This is a new milestone based on our experience and recent research directions.)

v. Applications:
- Evaluate methods for scheduling workflow applications on large-scale TIGRE grid
  (This is a rewording of previous milestones.)
- Explore additional TIGRE applications
  (This replaces the previous milestone "Extend application research as suggested by year 3 and 4 experience".)

vi. Education, Outreach, and Training:
- Continue AGEP program
- **Milestone moved to Year 4:** Participate in Grace Hopper Conference
  (changed due to schedule change for Hopper Conference)
- Continue graduate student exchanges

II. Findings

During the reporting period (6/1/06–5/15/07), VGrADS research focused on three inter-institutional efforts: *Applications, VGrADS Programming Tools,* and *VGrADS Execution System*. The following sections summarize the findings of each subproject.

1 Applications

LEAD: We successfully demonstrated the applicability of the VG abstraction and VGrADS scheduling techniques to the workflow from an important meteorological application. This can (in principle) address LEAD's requirement for transparent resource selection and monitoring. We believe, but have not yet verified, that this also provides a means to address LEAD reliability 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.

2 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.

3 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.

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 (6/1/06-5/15/07).

1 Training and Development Activities

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

1.1 Inter-institutional Collaboration

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

An important part of this interaction was student attendance at small-group workshops. The demonstration (at SC'06) of integrated VGrADS / LEAD execution led us to emphasize these meetings rather than student exchanges as we have done in years past. The small workshops are truly working meetings, producing detailed plans and software artifacts for use in our experiments and demonstrations. The first working meeting was held at UNC in April 2006, and resulted in technical advances and a detailed work plan for producing the VGrADS/LEAD integrated system demonstrated at SC'06. The students involved, of course, received significant experience in collaborative work, distributed software development, and a broader exposure to the project than they would ordinarily have had.

1.2 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 Fall 2006 semester, Lennart Johnsson taught *COSC6376: Grid Computing* at UH. For more information on this course and its contents, visit http://www2.cs.uh.edu/~johnsson/cosc6376_06/. In the Spring 2007 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/cs594-2007.htm. In addition, Lennart Johnsson is one of the organizers for the 5th International Summer School in Grid Computing (ISSGC '07), which will be held outside Stockholm, July 8–20, 2007.

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 2006, we sponsored three female students: Amanda Burch, Emily de la Garza, and Danielle Kramer. Amanda, a junior undergrad at Carnegie Mellon, researched known code optimization techniques and how they apply to the Play Station 3's new architecture. Specifically, Amanda tested the application of a loop fusion

algorithm to program code on Cell to lessen cache misses and thereby improve overall performance. Her work was done under the supervision of Dr. Ken Kennedy. Emily, a junior in the computer science program at the University of Houston-Downtown, used the EMAN program to compare and contrast the use of batch queues with performance models and Condor. Danielle, a sophomore in computer science at Carnegie Mellon, studied the effect of inaccurate cost estimates on various Grid scheduling algorithms. Emily and Danielle worked under the direction of Dr. Chuck Koelbel (who also shared supervision of Amanda). Emily, Danielle, and Amanda were provided with Grid-related reading materials in advance of the summer, which helped them prepare for their summer research activities.

Recruiting for the summer of 2006 is nearly complete. Approximately 100 students have applied to the AGEP program, of whom more than a dozen have specifically expressed interest in working with VGrADS. Two students from University of Houston Downtown, a local minority serving institution, have accepted our invitation for the summer. One of these students is Emily de la Garza, who will continue her work from last summer; the other is Lauren Garcia, whom Emily recommended to AGEP and VGrADS. Other offers are also outstanding. As we did last year, we will provide each student with Grid-related books to jump-start the research process. 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 2006, leveraging support from Rice's Dean of Engineering and the Department of Computer Science, VGrADS was able to provide travel support for the three 2006 VGrADS summer students, ten other female computer science graduate students and undergraduates, and one VGrADS staff member to attend the Grace Hopper Celebration of Women in Computing 2006 (http://gracehopper.org/2006/).

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/) will be held October 17-20, 2007, and will be co-located in Orlando, Florida with the Richard Tapia Celebration of Diversity in Computing conference (http://www.richardtapia.org/2007/), which is scheduled for October 14-17, 2007. The two conferences will share a day when attendees from either conference can attend joint sessions. During our next funding period, VGrADS plans to support female and/or minority students to attend both conferences. Dr. Charles Koelbel, a VGrADS PI, is serving as Posters Chair for the Tapia Conference. We have also encouraged local students, both within VGrADS and outside, to submit to both conferences.

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. Richard Tapia gave presentations at CS-CAMP sessions in 2006 for high-school CS teachers and high-school girls. An extension of the program, designed for middle school girls, is planned for the summers of 2007 and 2008. VGrADS PIs Keith Cooper, Richard Tapia, and Chuck Koelbel have been involved in planning the new program, and will participate in sessions during the camp.

2.4 Computer Science Community Interactions

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

The VGrADS project had a variety of professional outreach activities at the annual SC'06 Conference (Tampa, FL, November 11-17, 2006). Other sections of this report detail the research advances that were reported at SC'06, including paper and poster presentations. VGrADS researchers also gave a number of talks with accompanying demonstrations in exhibit booths, including:

- Ken Kennedy (Rice) gave many overviews of the VGrADS project at the GCAS booth (a collaboration of Rice, UH, and Texas A&M), the USC/ISI booth, the SDSC booth, and the RENCI (UNC) booth.
- Lavanya Ramakrishnan (IU, formerly UNC) led a team of students and staff in demonstrating the LEAD application running with VGrADS scheduling on batch queues and advance reservations.
- Dan Nurmi presented a paper (with A. Mandal, J. Brevik, C. Koelbel, K. Kennedy, and R. Wolski) on "Evaluation of a Workflow Scheduler Using Integrated Performance Modelling and Batch Queue Wait Time Prediction".
- Yang-Suk Kee presented a paper (with K. Yokum and A. Chien) on "Improving Grid Resource Allocation via Integrated Selection and Binding".

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

2.5 Other PI Involvement in Education and Outreach Activities

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

- Keith Cooper (Rice) is the PI on a grant from the Luce Foundation to fund Clare Booth Luce Fellowships in the Computer Science Department at Rice. The intent of this program is both to increase the pool of female applicants to the department's Ph.D. program and to provide a generous stipend ($25,000) for recipients. A two-year fellowship was offered in Fall 2005, Fall 2006, and Fall 2007.
- Lennart Johnsson (UH) has presented Grid-related talks at various venues including:
    - Information Management Forum, Houston, January 24th, 2007
    - "Experiences with Itanium Clusters in Grids", Gelato ICE Conference & Expo, Singapore, October 1 - 4, 2006
    - "Application Development and Execution on Grids", National Grid Singapore, Singapore, October 2, 2006
    - "Research and Education with and in State-of-the-art Networking", LEARN, September 6, 2006
    - "Grids and Medical Imaging", Institute for Biomedical Imaging Science Workshop, Kiawah Island, SC, August 10, 2006
    - "Grids: Enabling 21st Century Science and Engineering", Keynote, Baltic Grid Summer School, Tartu, Estonia, July 4 - 8, 2006
    - "Grids", HP-CCN Tutorial, Seoul, Korea, May 7, 2006
- Dan Reed (UNC) gave a talk in Second Life on the future of the 3-D net, sponsored by the New Media Consortium, which focused on educational uses of new technologies. See http://www.renci.org/blog/entry_18.php for more information.

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 this area include the introduction of slotted virtual grids, which allow unified management of reserved resources and resources controlled by batch queues. We construct slots (virtual reservations) for batch-queued resources by leveraging accurate predictions of the queue wait times; of course, reserved resources need no special mechanism to construct a slot. We have found the virtual reservation mechanism to be extremely useful on systems where we do not have administrative access.

  We have also added support in vgES for the concept of "resource equivalence", by which an application can identify trade-offs between processor architectures. For example, the EMAN application can state that Opteron processors run twice as fast as Itanium processors with the same clock speed (for this application). This allows additional flexibility in resource allocation and scheduling.

- VGrADS researchers have also begun to unify the virtual grid treatment of fault tolerance. This includes extensions to the Virtual Grid Description Language (vgDL) to express reliability requirements and to query reliability estimates for resources. We are using these extensions to evaluate scheduling mechanisms that incorporate reliability as a criterion, and to study support for fault tolerance mechanisms.

- VGrADS researchers are continuing the development of a temporal reasoning framework to support performance validation of virtual grids. In particular, this reasoning framework will support monitoring of grid applications, allowing them to identify changes in conditions that require adaptation.

- VGrADS researchers developed 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. This method obtains very good overall application

performance. We have proposed effective methods for choosing the virtual grid based on processor equivalence, basically rating each cluster according to its effective total computational power. We have also evaluated a number of scheduling heuristics for a virtual grid, and are developing improved heuristics based on those studies.

- VGrADS researchers continue to extend the range of applicability of a priori schedulers for computational Grids. We have shown that it is possible to combine estimations of application performance and batch queue wait times to generate high-quality schedules. We have also developed scheduling methods for slotted virtual grids, in which the resources are not simultaneously available. We have also developed schedulers that take into consideration robustness of the resulting schedule against performance anomalies, and reliability of the resulting schedule in the presence of processor faults.

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

- VGrADS researchers have developed the FT-MPI library to provide process-level fault tolerance based on the MPI 1.2 standard, 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.

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 they 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 Fall 2006 semester, Lennart Johnsson taught *COSC6376: Grid Computing* at UH. In the Spring 2007 semester, Jack Dongarra taught *CS 594-004: Scientific Computing for Engineers* at UTK. In addition, Lennart Johnsson is one of the organizers for the 5th International Summer School in Grid Computing (ISSGC '07).

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

VGrADS researchers have presented VGrADS research results and ideas in a variety of forums, including technical computer science presentations, presentations to applications groups, and presentations to students. In particular, the VGrADS project was involved in a variety of outreach activities at the SC'06 conference in Tampa, FL (November 11-17, 2006). VGrADS activities at SC'06 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.