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 have obscured the Grid's potential. The six-year Virtual Grid Application Development Software (VGrADS) project attacked a fundamental part of this problem—how to more effectively program these highly complex and dynamic systems. It developed software tools and methods to simplify and accelerate the development of Grid applications and services without compromising high levels of performance and resource efficiency. This improved usability should greatly benefit the community of Grid users and developers. In the process, VGrADS contributed to both the theory and practice of distributed computation.

To address these aims, VGrADS explored, defined, and implemented a hierarchy of virtual resources and a set of programming models for Grid computing. Its research covered three key areas:

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

VGrADS pursued this agenda by collaborating with leading scientific applications to elicit key challenges, validate results, and disseminate technology. It is also built on its PIs' past successes in human resource development by leveraging existing programs to attract and retain women and minorities in computational science.

Over its lifetime (10/1/03 – 9/30/09), VGrADS research focused on the three inter-institutional efforts described in the following sections: and *VGrADS Execution System (Section 1), VGrADS Programming Tools (Section 2),* and *Applications (Section 3).* Project publications and additional information can be found at http://vgrads.rice.edu. The project Web site includes a participants-only area where VGrADS researchers exchange drafts of project documents and other materials. The management structure of the VGrADS project is described in *Management & Structure (Section 4).* Annotated VGrADS milestones appear in *Project Milestones (Section 5).* We finish by presenting the *Graduate Student Thesis Abstracts (Section 6)* completed with VGrADS support.

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

As in past reports, our execution system research addressed the development of vgES (Virtual Grid Execution System) to support complex, adaptive workflow applications. Section 1.1 describes the basic capabilities of vgES, which might be considered the software infrastructure for VGrADS. Other sections expand on research that extends parts of this system.

As it was the last year of the grant, little "new" development occurred in vgES. Our efforts were spent in capitalizing on past success, and providing more robust software for our final experiments. One exception to this was the improved support for "cloud computing", which we began late last year and completed this year. Section 1.2 describes these efforts, and more details are available in a paper that appeared at CCGrid 09 and in our upcoming SC09 paper.

As this is a final report, we should note that some major efforts over the lifetime of the project came to fruition before our final year. We refer the reader to previous annual reports for descriptions of such research directions as temporal reasoning for resource characterization (2008 report), Virtual Advanced Reservations for Queues (VARQ) (2007), monitoring the virtual grid (2006), and simplifying grid scheduling (2006). Each of these areas led to at least one thesis during the course of the project.

In addition to the above, we have conducted research in the four areas described in detail in the indicated sections: slot allocations using both advanced and virtual reservations (Section 1.1.2), grid computing using local cluster resources (Section 1.2), fault tolerance in the Virtual Grid (Section 1.3.1), and tuning parameters for fault-tolerance methods (Section 1.3.2).

As it has been for the past several years, vgES formed the basis for a VGrADS demonstration at the annual SC conference (SC08 in Austin). 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.

1.1 vgES Fundamentals

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

1.1.1 A Generic Framework for Resource Allocation and Execution

Since many Grid resource managers were designed for time-sharing or dedicated resources, the resource acquisition process has been ignored or, at most, been simple

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

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

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

1.1.2 Slot Allocation over Advance Reservation and Virtual Reservation

As VGrADS developed, our concept of the resources needed for an application did as well. By then end of the project, we had settled on the construct that we termed a *slot* as the basic unit of scheduling. A slot is a high level representation of resources in time and space. It is a set of resources available in a certain time range. It can be expressed by a tuple <size, start time, duration>. Different from the slot used in a batch system that describes one resource allocation across time and space for the single resource from the system viewpoint, the slot in vgES is a resource collection in time and space across multiple resources from the application viewpoint.

We implemented this slot concept against advance reservations provided by some batch systems and virtual reservation using batch queue prediction, as detailed in previous annual reports. This year, we added implementations for cloud computing systems. In effect, both traditional batch-queued systems and cloud systems were managed by the same VG interface, which translated our commands into the format (queue submissions or image start-ups) appropriate to the underlying cluster or cloud. The resulting system was highly flexible, and provided an excellent foundation for our demonstrations at SC08 in Austin and our SC09 paper experiments.

1.2 Cloud Computing Local Cluster Resources

During the past two years, the VGrADS team at UCSB developed EUCALYPTUS – *A*n *E*lastic *U*tility *C*omputing *A*rchitecture for *L*inking *Y*our Programs to *U*seful *S*ystems. Eucalyptus is an open-source reverse-engineered implementation of Amazon's EC2 cloud computing infrastructure that can be deployed on local cluster resources. Using SDSC Rocks as a deployment tool, any site installing Eucalyptus can offer cloud-computing services from its resources using the user-interface and command-line tools available from Amazon.com. It is EC2@home.

The intention was to include cloud computing in the portfolio of systems that vgES can amalgamate automatically beneath the VGrADS slot abstraction. Various VGrADS sites (UCSB, RENCI, UH) have run Eucalyptus, each exporting its own cloud, which was then integrated into a single set of slots by vgES. In addition, because Eucalyptus uses the Amazon.com interface tools, integrating Eucalyptus with vgES allowed VGrADS to also use Amazon's EC2 cloud (although there was an occupancy charge).

Eucalyptus has had a wide impact outside VGrADS itself. While part of VGrADS, Eucalyptus was downloaded approximately 17,000 times in more than 80 countries. It is now being used on every continent except Antarctica (and we are hopeful there as well). It was integrated into the Ubuntu Linux release, and has been part of every copy of that system since Ubuntu 9.04 server edition. The Ubuntu release has led to more than 20,000 additional downloads. In 2009, VGrADS PI Rich Wolski co-founded Eucalyptus Systems, Inc. to provide support for Eucalyptus. While the software remains open-source, the new company provides on-premise private and hybrid cloud computing solutions for large-scale enterprise deployments. As Wolski said at the time the company was formed, "Eucalyptus Systems will ensure the viability and growth of Eucalyptus well beyond its life as a university research project, while also extending the technology to meet the needs of organizations that require high scalability, reliability, and enterprise-grade support." All this has led to excellent technology transfer from this VGrADS project.

1.3 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 were tools and mechanisms to monitor performance and ensure reliability (e.g., via replication and over provisioning, checkpoint/restart and other schemes), few tools allowed users to express reliability policies from the application's perspective, map these to resource capabilities, and then coordinate and enforce strategies. We filled this gap by implementing additions to vgES as described in the subsections below.

1.3.1 Fault-tolerance for slots

During VGrADS Year 6, we integrated algorithms for slot replication with workflow engine, which is the planning and execution engine in the VGrADS software stack. The core slot-replication algorithm was developed last year, while the integration with the workflow engine was accomplished this year. These algorithms provided automatic replication capabilities to the workflow planner.

The workflow planner interacts with a fault tolerance component to determine if a task should implement replication in the second case. For this implementation, the fault tolerance component implements replication based fault-tolerance techniques to increase the probability of success for each workflow task. Given the current mapping of tasks on a Gantt chart of available resource slots with corresponding reliability characteristics, each workflow task is replicated on additional slots to increase the probability of success of a task to the desired success probability. The fault-tolerance techniques determine the mapping of the replicated tasks on the available slots and return the mapping to the planner. During this mapping process, we use simple techniques based on joint probabilities derived from success probabilities of tasks on slots. For each task, a window of replication is determined by the planner, which constrains the replicated tasks to start-after and end-before particular time based on task dependencies. The fault tolerance mapping process tries to fit a task on the available slots in that replication window based on the expected performance characteristics of the task (number of CPUs required and expected execution times derived from performance models). If the task fits on a slot, the success probability of the task increases. When the success probability reaches the desired level during this replication process, the mapping of the replicated tasks is returned to the planner.

We demonstrated fault-tolerance on slots with LEAD workflows during the VGrADS presentations at the SC08 conference in Austin. Also included in that demonstration were some tools advances that are described in other sections.

1.3.2 Experiments with fault-tolerance parameters

We also designed a set of experiments to explore the different parameters that affect the median replication factors and replication success rate for providing fault-tolerance on slots -- reliability increments, underlying slot reliability etc. The results of these experiments have been included as a section in a paper published at the Supercomputing'09 (SC) conference.

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

The broad vision of the programming tools thrust was 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 was our attempt to take advantage of the virtual grid (VG) abstraction and tools to provide more application-specific abstractions.

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

1. Scheduling for workflow computations taking queue delays into account,
2. Scheduling workflow computations taking fault tolerance into account,
3. Fault-tolerant libraries for MPI,
4. Resource allocation via grid economies, and
5. Optimization of FFT routines,

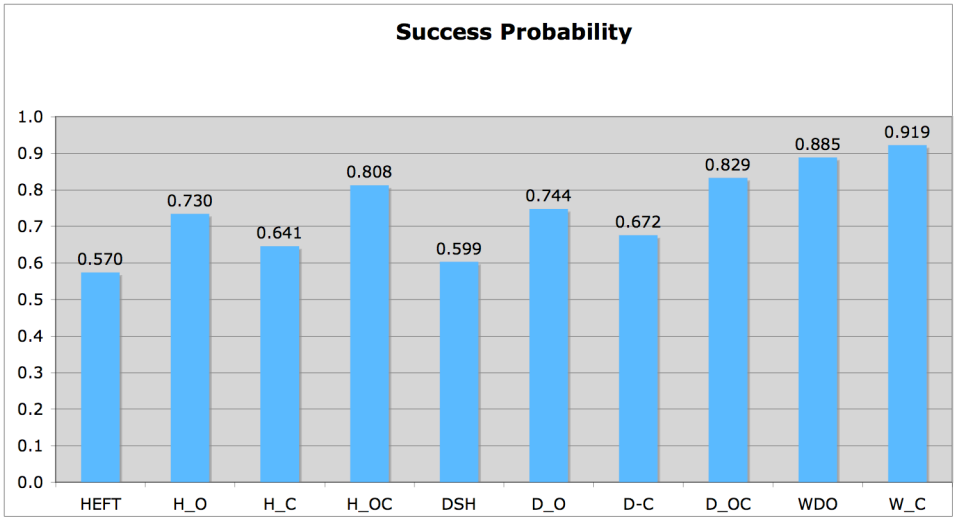
The following subsections discuss each thrust in turn.

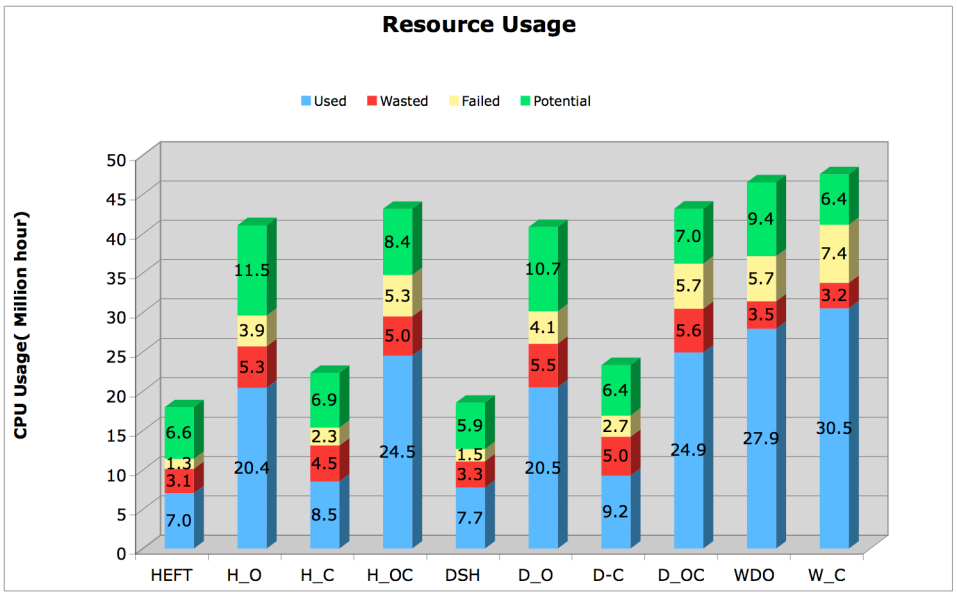
Some major programming tools projects came to fruition before the end of the grant. We refer the interested reader to our previous annual reports for such topics as resource allocation via grid economies (2008 annual report) and compiling and optimizing node programs (2007), both of which led to theses. In addition, several other variants of scheduling for grids led to thesis work in previous years, which is reflected in our cumulative publication history.

2.1 Scheduling Workflows for Fault Tolerance

Rice and UNC collaborated in proposing new scheduling approaches that combine fault tolerance techniques (overprovisioning (replication) and checkpoint-restart) with existing workflow scheduling algorithms. We also studied a novel scheduling algorithm that performed overprovisioning at a whole-application level, as opposed to the previous approaches that replicated at a single-task level. UNC was responsible for development and integration of the fault-tolerance algorithms component, while Rice was responsible for combining these with workflow scheduling algorithms and experimentations.

This work has been published at the CCGrid'09 conference. In this paper, we present a study on the effectiveness of the combined approaches by analyzing their impact on the reliability of workflow execution, workflow performance and resource usage under different reliability models, failure prediction accuracies and workflow application types. The key finding was that we could quantify, to some extent, the trade-offs between reliability, performance, and resource usage. Two charts show part of this story:



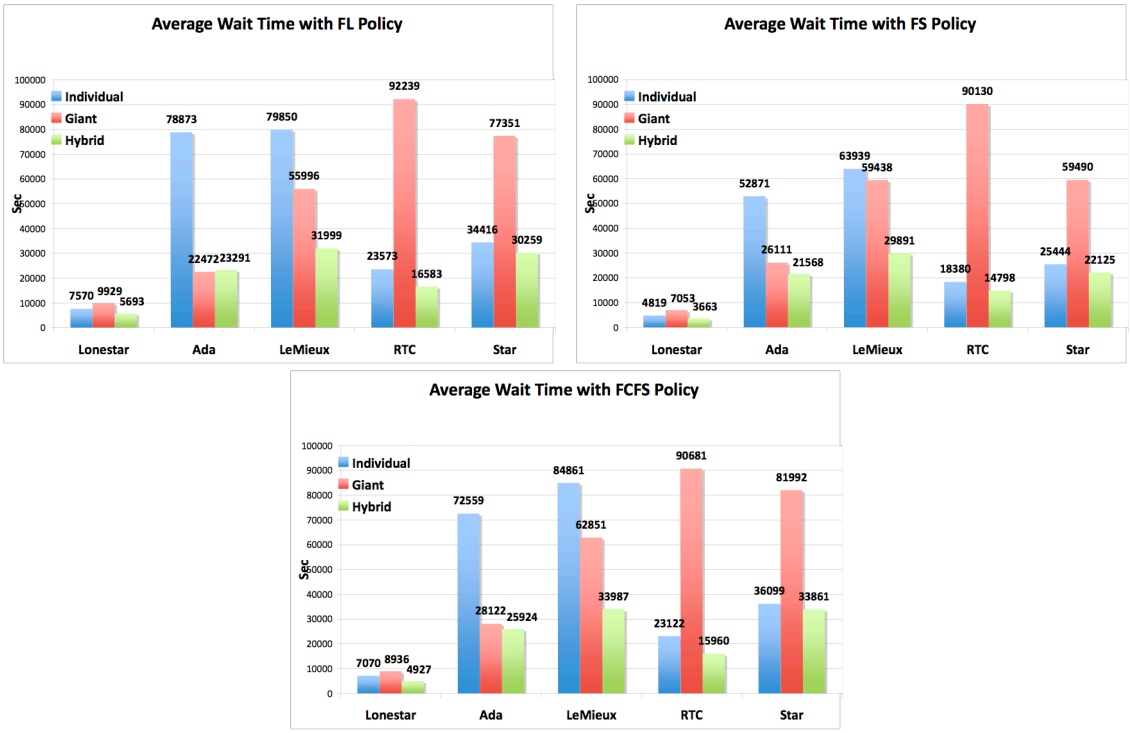


As the graph of success probability shows, node-based overprovisioning – signified by "_O" in the method name – improves reliability by about 25% while node-based checkpointing – signified by "_C" – only adds 12% to reliability. Whole DAG Overprovisioning (WDO) does best of all by this measure. Of course, this comes at a cost, as seen in the resource usage graph. Overprovisioning uses more than twice the actual CPU time as the non-fault-tolerant version of the same method, while checkpointing adds 50% overhead. WDO, again, is the extreme case, using the most CPU time.

2.2 Scheduling Workflows with Queue Delays

Another VGrADS paper, presented at Cluster2009, considered workflow scheduling onto batch queue controlled clusters. Although vgES makes it possible to schedule onto grids of such clusters, this work addresses single-cluster execution. Because inter-cluster communication costs often dominate the computation, this is an important special case even for grid computing. Our new method uses runtime monitoring of application performance (including computation, communication, and file I/O time) and estimated wait times from the batch queue system itself to appropriately partition the workflow DAG for execution. The key insight was that, by overlapping one job's execution with another's queue wait time, we could hide much of the total waiting time. This could, of course, be taken too far – submitting every individual task as a separate job produces jobs that are too short to effectively hide waiting times. We developed a "peeling" algorithm to move DAG levels from one job to another to balance the overlapping and wait times.

Based on extensive simulations (over 700,000 individual experiments), we were able to show that our hybrid method outperformed the most common workflow scheduling methods, either submitting the entire task as a single batch queue job (the "giant" method in the charts below), or submitting every task as an individual job ("individual" below). Our basic results are captured in the three charts below.



Significantly, our hybrid scheduling method is the best method under variety of load conditions (competing with actual queue logs from five clusters). It is also robust to

queue policies of favoring long-running jobs (FL), short-running jobs (FS), and first-come, first-served (FCFS).

This work and the work in Section 2.1 are a major part of Yang (Ryan) Zhang's Ph.D. dissertation, which should be accepted by the time this report is submitted.

2.3 Fault tolerant MPI: FT-MPI / OpenMPI

The OpenMPI project has been combining the 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. Recent efforts at UTK include improving scalability and fault tolerance in OpenMPI.

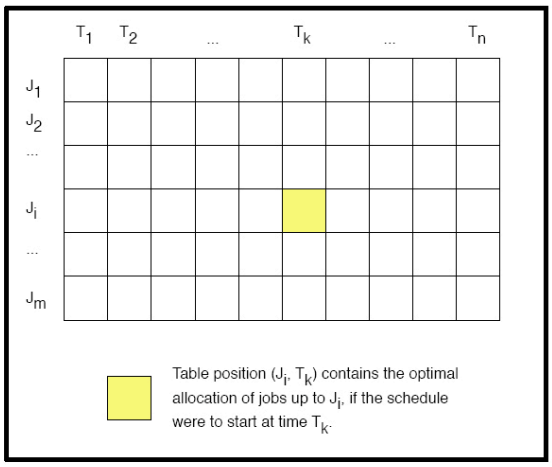
In order to support fault tolerant MPI applications, we require a fault tolerant runtime environment under the MPI library. A fault-tolerant runtime environment must detect failures, and coordinate with the application to recover from them. Previous work has demonstrated that the Binomial Graph topology (BMG) is a good candidate as a communication infrastructure for supporting both scalability and fault-tolerance for runtime environments. Recently, we have designed and analyzed a self-stabilizing algorithm to transform the underlying communication infrastructure provided by the launching service into a BMG, and maintain it in spite of failures. We demonstrate that this algorithm is scalable, tolerates transient failures, and adapt itself to topology changes (Bosilca et. al. ParCo2009)

2.4 Resource Allocation via Grid Economies

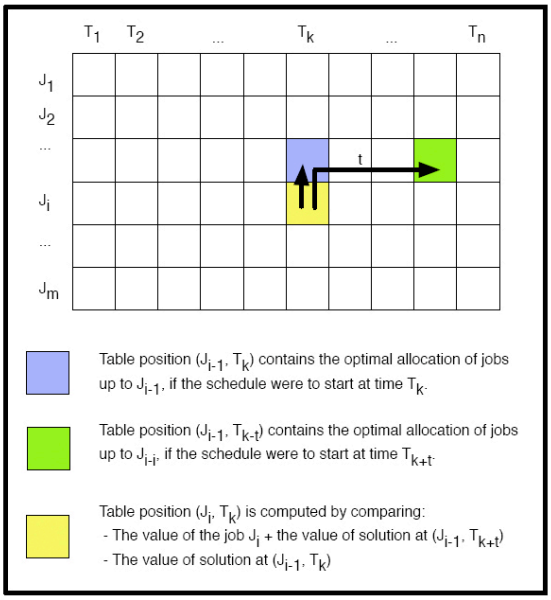
As reported in previous annual reports, researchers at UCSB have embarked on a study of economics-based methods for resource allocation and scheduling on grids. The inspiration for this study was Ken Kennedy's observation that application performance models (also developed under GrADS and VGrADS) provide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

Our approach to this idea is the development of auction mechanisms for control of batch queued resources. The key problem in this space is extracting accurate information about the performance and importance of the individual jobs. If users believe that they can improve their private results (e.g. shorten their own wait times) by lying about such matters (e.g. underestimating resource usage), they have an incentive to do so. If many users react in this way, the queueing system will base its scheduling on false information, resulting in sub-optimal global performance. Carefully designed auctions to "buy" the batch queue slots can avoid this pathology by ensuring that each user's best strategy is to bid honestly; underbidding will result in not acquiring enough resources, while overbidding will cost the user too much (and thus impoverish him or her in future bids).

We have previously reported on several papers by Andrew Mutz, a student working under VGrADS PI Rich Wolski, who has been investigating this problem. During VGrADS Year 6, Mutz presented a paper at SC08 entitled "Efficient Auction-Based Grid Reservations using Dynamic Programming" in which he demonstrated practical methods integrated into the Portable Batch System (PBS) implementing a form of Generalized Vickery Auction (GVA) for scheduling. Previous papers have demonstrated that while GVA with combinatorial bids (i.e. acquiring sets of resources) is optimal, computing the necessary payments is an NP-complete problem. Mutz, in contrast, accepts simpler bids consisting of the job value (*v*), run time (*t*), and deadline (*d*). He then constructs a dynamic programming algorithm that computes the optimal schedule (runs the jobs with the highest total value possible) and required payments (resolving the auction bids). The diagram below illustrates the algorithm; cells are computed from right to left, and from the top of the diagram down. Other results in the paper show that this is orders of magnitude faster than computing full combinatorial GVA prices.



The structure of the table that stores intermediate results



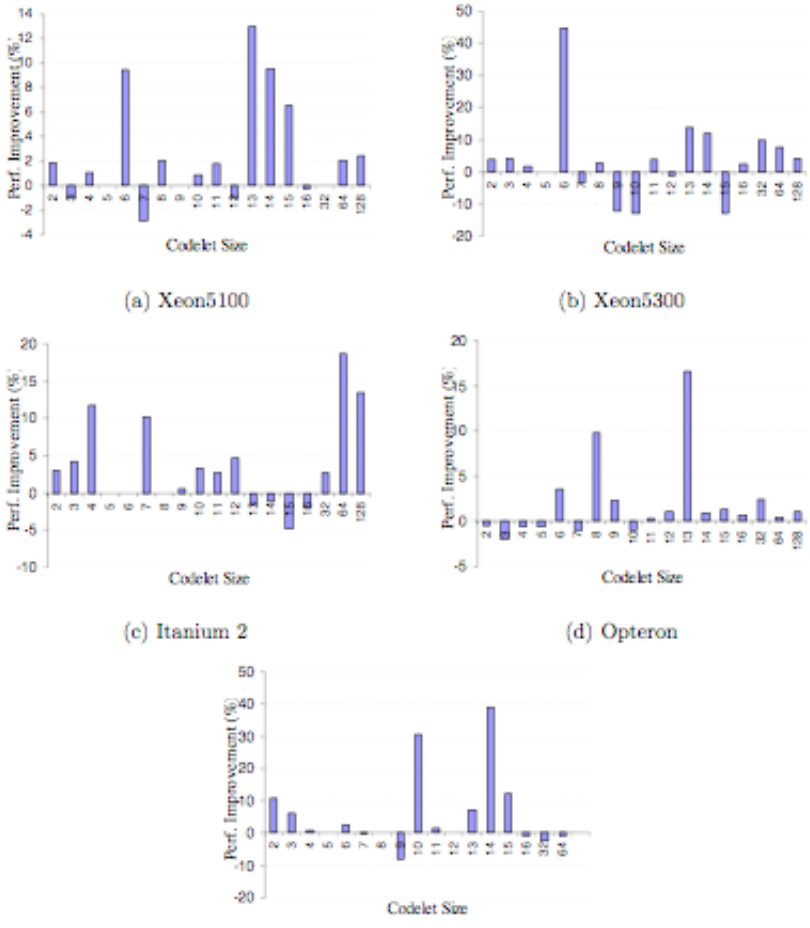
The recursion relation. The yellow cell is computed from the blue and green cells. The table is filled from the top right.

This work, as well as previously reported work on grid economies, was also incorporated in Mutz's Ph.D. thesis.

2.5 Optimization of FFT Routines

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

Ayaz Ali, a student of VGrADS PI Lennart Johnsson, has published a series of papers and a thesis about compilation of the Fast Fourier Transform algorithm for modern computer architectures. Because the FFT uses very particular data access patterns and non-nearest-neighbor communications, many compiler techniques developed for linear algebra and mesh relaxation do not produce good results. However, the variety of implementation options available (corresponding to recombinations of the Kroenecker product formulation of the FFT) allows many opportunities for optimization. In his thesis, Ali presents a code generator framework and search strategy for finding the best series of recombinations. As the graph below (taken from the thesis) shows, the optimal parameters chosen for a given architecture can vary significantly, as can the size of the improvement by choosing an optimal parameter value.



(a) Xeon5100 (b) Xeon5300

(c) Itanium 2 (d) Opteron

(e) Power5

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

VGrADS research has always been driven by the needs of actual applications. Initially, we selected four applications (EMAN, EOL, GridSAT, and LEAD) based on our experience in the GrADS project and from other sources to help derive requirements for Virtual Grid (VG) functionality and to serve as tests of new tools methods. We completed our study of EOL – the Encyclopedia of Life – in 2005, having learned invaluable lessons about application structure from it (but ultimately concluding that integration with vgES would be infeasible). Our work on EMAN – Electron Micrograph Analysis - was substantially complete by 2007, but we continue to use its DAGs as test cases for new scheduling methods. GridSAT – solving SATisfiability problems on the Grid –was arguably our most successful application, having been the first program to solve a number of provably hard (and at the time unsolved) logical problems. It was the basis for many of the basic requirements for vgES and some excellent scheduling work. Work on this program was complete in 2006, when Walid Chrabakh defended his dissertation. We refer the interested reader to previous annual reports for details on these applications. The remainder of this section discusses the LEAD application in more depth (Section 3.1), as well as three new applications that were developed during VGrADS. GridSolve (Section 3.2), a grid-enabled system for mathematical computations, arose from study of fault tolerance mechanisms and UTK's long-standing interest in mathematical software. Similarly, we have developed applications of large-scale linear algebra (Section 3.3) and fault-tolerant MPI (Section 3.4) in response to needs in those communities. To date, these applications have been reasonably successful at helping to derive requirements for the VG functionality and vgES implementation.

3.1 LEAD

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

The unique characteristics of LEAD lie in the dynamic workflow orchestration and data management, which allow the use of analysis tools, forecast models, and data repositories not only in the current fixed configurations, but as dynamically adaptive, on-demand, Grid-enabled systems that can (a) change configuration rapidly and automatically in response to weather; (b) continually be steered by new data; (c) respond to decision-driven inputs from users; (d) initiate other processes automatically; and (e) steer remote observing technologies to optimize data collection for the problem

at hand. Toward these goals, LEAD research 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 collaborated with the LEAD researchers and developers to develop integrated plans for the two projects. This process was greatly expedited by the efforts of Lavanya Ramakrishnan, who began as VGrADS programmer at RENCI and recently received her Ph.D. with LEAD PI Dennis Gannon, and Anirban Mandal, who started as a Ph.D. student at Rice and eventually became the VGrADS PI at RENCI. They organized a number of meetings and teleconferences, and generally led the development of the LEAD/VGrADS integration.

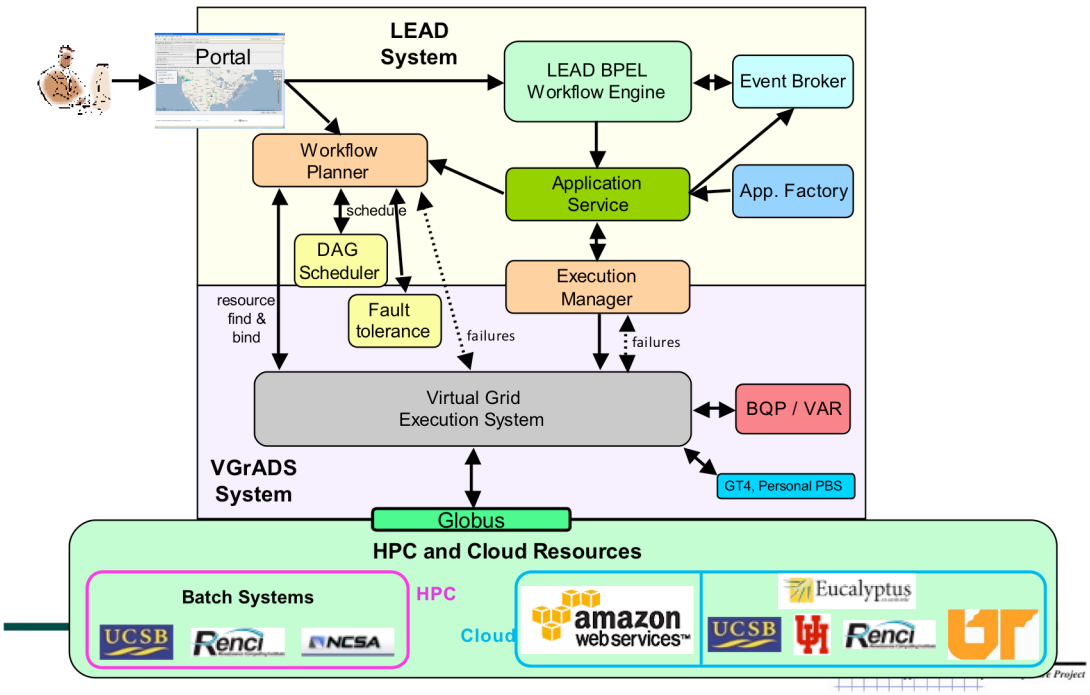
As noted in previous annual reports, we have had meetings with LEAD personnel to map out a collaborative strategy. A number of technical issues had to be resolved, mostly relating to how VGrADS would interoperate with LEAD's existing software infrastructure. Also, we had to address the fact that LEAD's definition of workflow is based on the BPEL web-service workflow specification that differed from other VGrADS applications in implementation details. Every component in the LEAD architecture is encapsulated as an individual web service that represents the atomic application tasks and the resource and instrument monitoring agents that drive the workflow. Other VGrADS applications do not use such services.

During VGrADS Year 6, we built on our successful incorporation of fault-tolerance and runtime adaptation functionalities into the VGrADS/LEAD integration software stack to enable resilient execution of LEAD workflows. Resilient workflow execution was of paramount importance to LEAD workflows because of deadline constraints (forecasts have to be available by a certain deadline). We had demonstrated these resilience features previously at SC07. Following this up for our demonstration at SC08 in Austin, we incorporated vgES control of cloud computing into the process, allowing what we believe was the first demonstration of simultaneous use of TeraGrid, Amazon EC2, and local (batch-controlled clusters and cloud systems based on Eucalyptus) resources for a workflow set i.e., collection of workflows. This work was a true collaboration, involving Lavanya Ramakrishnan (Indiana University), Anirban Mandal (RENCI), Gopi Kandaswamy (RENCI), Dan Nurmi (UCSB), Kiran Thyagaraju (Rice), and many others. We should particularly mention Yang-Suk Kee, who had been a long-time VGrADS participant at UCSD and ISI; early in the process, he left the project for a new position at Oracle. Before leaving, however, he successfully developed the necessary interfaces from vgES to EC2 and Eucalyptus clouds, enabling the project to succeed. Far from a simple port of the vgES interface, this required significant discussion of the semantics of "binding" and "virtual reservation" to a cloud and other deep technical issues. . In addition, the vgES interface was expanded to represent the resources returned to the application layer as a Gantt Chart (i.e., set of resource slots), a format

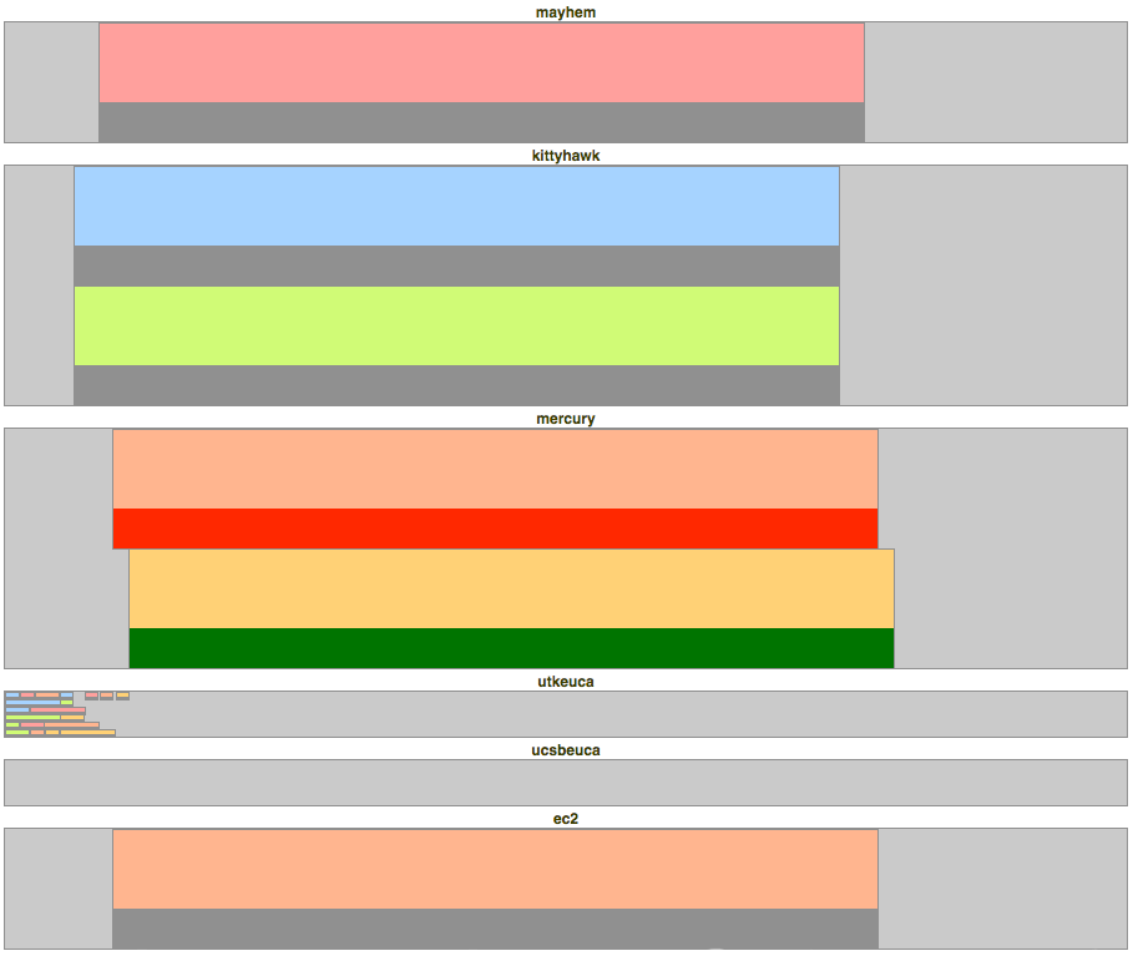
used by LEAD workflow orchestration to make scheduling decisions. This enabled the workflow orchestration algorithms in LEAD to seamlessly use VGrADS interfaces.

The figure below (taken from our demo presentation) shows the relevant components of the VGrADS/LEAD integration stack and the invocation flow for various components. The overall system used nearly every technology that VGrADS had developed over its lifetime. Few changes were needed in LEAD itself; in essence, its Execution Manager and Workflow Orchestration system required new interfaces to vgES (where previously they had directly invoked Globus or the individual systems). However, the interface changes allowed the Workflow Orchestration to make application-specific tradeoffs. In particular, LEAD requires that a certain percentage of the given workflows complete by a given deadline. Given information from vgES, the planner uses a four-phase orchestration method to balance the need for performance (finishing before the deadline), quality (completing as many workflows as possible), and reliability (completing enough tasks). Within VGrADS, we used our Batch Queue Prediction (BQP) and Virtual Advanced Reservation (VAR) technology to schedule slots on batch queue controlled resources. When it was profitable, we started images on cloud (Eucalyptus and EC2) resources to create slots there. Of course, Eucalyptus itself was a new system developed as part of VGrADS. The Fault Tolerance and Recovery (FTR) system from RENCI was used to determine the amount of task replication needed to achieve a given reliability, and to recover tasks that failed. The workflow actually executed on machines at four VGrADS sites (UH, UTK, UCSB, and RENCI), one TeraGrid site (NCSA), and one commercial cloud computing site (Amazon EC2).

LEAD / VGrADS Architecture



Below is a snapshot from the demonstration. The LEAD requirement was that one workflow (out of eight submitted) must complete within 2 hours. Based on the resources reported by vgES, five were scheduled across six sites. The slot at each site is represented by the background gray box, and individual tasks within the workflow by colored boxes (one color per workflow). The small tasks on UTKEuca (the Eucalyptus machine at UTK) are serial preprocessing tasks that could start immediately; the larger tasks on other machines are the actual weather simulations running on larger clusters. Here, one salmon-colored task was replicated (on Mercury (NCSA) and EC2) for reliability. The bar at the bottom of a task represents its status when the snapshot was taken – green for running, dark gray for complete, and red for failed. In this case, one workflow (at Mercury) failed but its copy succeeded. In short, vgES helped ensure that the system could more than meet the demands of the application in this case.



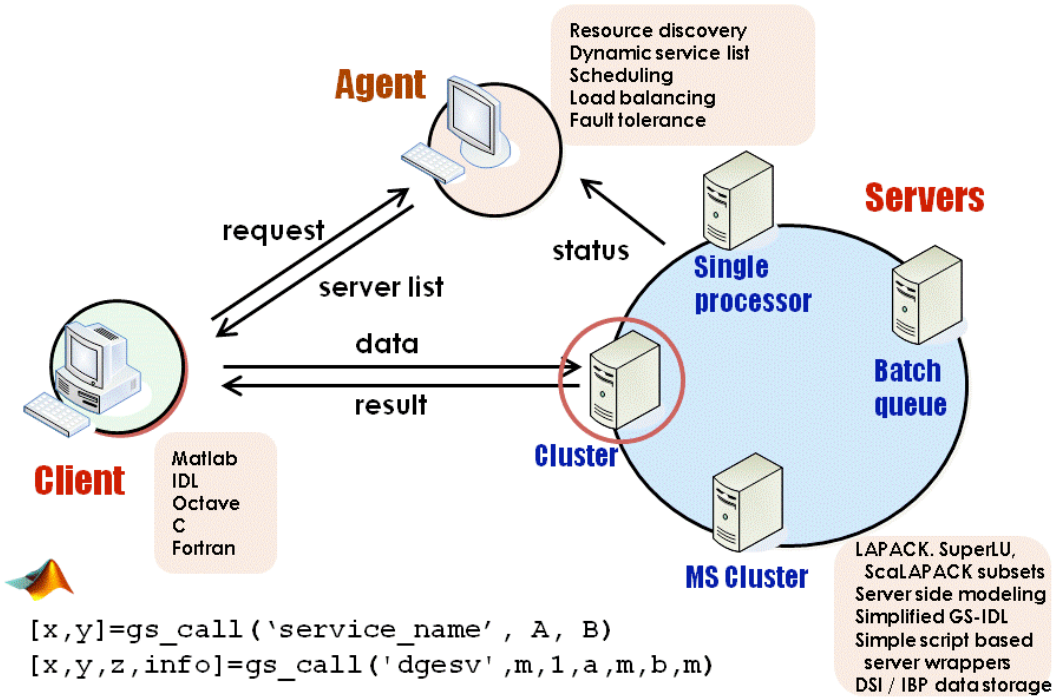
The failure in the example above was neither planned, nor a rare event. In fact, we had actual task failures in some of our live demos at SC08. However, it is a testament to vgES that none of these stopped the application, nor did any cause the experiment to miss its deadline.

The SC08 demonstration was described, along with quantitative measurements, in a paper accepted at SC09. In many ways, that paper is a fitting capstone to the development we have done for VGrADS.

3.2 GridSolve

The GridSolve project attempt to provide a seamless bridge between the between the standard programming interfaces and desktop systems that dominate the work of computational scientists and the rich supply of resources and services supported by the grid. Computational scientists using their accustomed interfaces (e.g., Matlab, Octave, IDL, C, Fortran) can easily access grid resources with low effort, by using the OGF standard gridRPC API in GridSolve. Transparent to the user, GridSolve provides

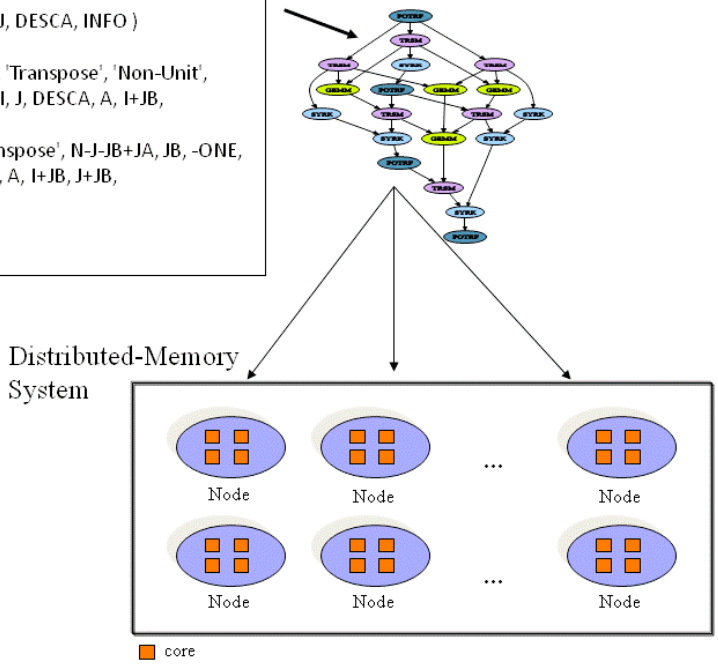
benefits like resource discovery, scheduling, load balancing and service level fault tolerance. During the last year, GridSolve has added workflow execution mechanisms while retaining the ease-of-use that is a main focus of the project. If a sequence of gridRPC calls to grid services is tagged as a workflow (using simple begin and end tags), GridSolve will transparently analyze the data dependencies between that service calls, infer a task DAG from the dependencies, and schedule and execute that task DAG. Inter-server data transfer and task scheduling are all handled by the workflow execution mechanism. Yinan Li, a graduate student under PI Jack Dongarra, reported on this work in "Request Sequencing: Enabling Workflow for Efficient Problem Solving in GridSolve" (Li et al GCC2008). Future work (supported by other sources) will focus on scheduling and mapping strategies for this workflow engine.



GridSolve has also integrated a second workflow execution mechanism as part of a collaboration with UC Dublin. SmartGridSolve (Brady et. al. 2008) allows tasks to be mapped collectively and supports the minimization of the execution time of a group of tasks collectively. SmartGridSolve uses a simple extension to GridSolve to delineate a sequence of service calls, then infers the task DAG from that sequence and executes the DAG. It has a strong focus on efficient mapping strategies designed to take into consideration the communication and communication to improve the performance.

3.3 Linear Algebra: Dynamic Task DAGs for Large-Scale Linear Algebra

```
SUBROUTINE PDPOTRF( UPLO, N, A, IA, JA, DESCA, INFO )
  DO 20 J = JN+1, JA+N-1, DESCA( NB_ )
     JB = MIN( N-J+JA, DESCA( NB_ ) )
     I = IA + J - JA
     CALL TPDPOTF2( UPLO, JB, A, I, J, DESCA, INFO )
     IF( J-JA+JB+1.LE.N ) THEN
       CALL TPDTRSM( 'Right', UPLO, 'Transpose', 'Non-Unit',
$                N-J-JB+JA, JB, ONE, A, I, J, DESCA, A, I+JB,
$                J, DESCA )
       CALL TPDSYRK( UPLO, 'No Transpose', N-J-JB+JA, JB, -ONE,
$                A, I+JB, J, DESCA, ONE, A, I+JB, J+JB,
$                DESCA )
     END IF
  20 CONTINUE
```



UTK has a long-term interest in providing high-performance mathematical solvers to enable computational science. To that end, we have been revisiting the approaches taken by the LAPACK and ScaLAPACK libraries. In the current approach, linear algebra algorithms are viewed as consisting of tasks that perform operations on tiles of data, with data dependencies between the tasks. The dependencies allow a task DAG to be inferred from the sequence of tasks. If the tasks in the DAG are scheduled appropriately, many of the serialization and synchronization points that were in LAPACK/ScaLAPACK can be hidden or avoided by overlapping tasks. Fengguang Song, a graduate student under PI Jack Dongarra, applies these ideas in the distributed setting by constructing and executing dynamic DAGs on distributed memory multi-core machines for linear algebra applications (Song et. al. SC2009).

3.4 FT-LA: Fault Tolerant Linear Algebra

UTK is continuing to explore scalable techniques to tolerate a small number of process failures in large-scale computing. The scientific community has to tackle the problem from two directions. First, efficient middleware needs to be designed to detect failures. Second, the numerical applications have to be flexible enough to permit the recovery of

the lost data. At UTK, we successfully developed a Fault Tolerant MPI (FT-MPI) middleware and more recently, a Fault Tolerant Linear Algebra (FT-LA) library that will efficiently handle several process failures. The FT-LA approach combines fault-tolerant techniques for extending the algorithms, with diskless checkpointing and efficient error detecting and correcting codes (Bosilca et. al. JPDC 2009). This work is being developed by UTK graduate student Peng Du under the direction of PI Jack Dongarra.

Our future work in this area (under separate funding) will involve the development of scalable fault-tolerant one-sided (Cholesky, LU and QR) and two-sided (Hessenberg, tri-diagonalization and bi-diagonalization) factorizations in the context of tile algorithms using task DAGs. In case of failures, the core idea would be to restart the computation by mostly using critical information already present in the directed acyclic graph generated by those factorizations, which will ultimately considerably decrease the checkpoint sizes.

4 Management & Structure

VGrADS includes researchers from Rice University; University of California, San Diego (UCSD); University of California, Santa Barbara (UCSB); University of Houston (UH); University of North Carolina (UNC); University of Southern California / Information Sciences Institute (USC/ISI); and University of Tennessee, Knoxville (UTK). Rice University serves as the lead VGrADS institution. Keith Cooper serves as VGrADS PI. He is advised by an executive committee, which consists of the key researchers leading the main VGrADS research thrusts. The current members of the VGrADS executive committee are:

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

During VGrADS Year 6, the VGrADS executive committee communicated by e-mail and teleconference to review progress and milestones, discuss plans for the future, and advise the PI on resource allocation issues. The most significant decisions were reallocation of resources as UCSD and ISI completed their work for the project and returned unspent funds. These were used to complete work on the final demonstrations reported above.

Project design and coordination during our sixth year were enabled through weekly technical teleconferences involving researchers from each of the VGrADS sites, one developers' workshop at USC/ISI (9/8-9/09), and communication via VGrADS mailing lists. Research subproject participants also met on a regular basis to exchange ideas and develop research plans.

4.1 VGrADS Web Site

During the life of the project, the VGrADS Web site, http://vgrads.rice.edu, was updated to reflect recent results, current project directions, and personnel changes. The web site will continue to be hosted indefinitely at Rice, although we will not actively update it after the end of funding.

5 Project Milestones

As recommended by the NSF Site Review team (4/28-29/05), the VGrADS Principal Investigators have actively tracked and (where necessary) updated research milestones for the project. Here we report on milestones in years 5 (some of which were still in

progress as of our last report) and 6. All section numbers refer to the 2009 report unless otherwise noted.

5.1 Year 5 Milestones

We have made excellent progress on Year 5 milestones. We summarize our activities thus:

i. Execution System/Virtualization:
   - Improve resource selection and binding techniques for flexible resource discovery.
     (Done. Described in 2007 and 2008 reports.)
   - Improve resource-scheduling techniques for VGs that consider resource efficiency and cost.
     (Done. Described in 2007 and 2008 reports.)
   - Prototype and evaluate extended VG abstraction over environments with diverse resource management paradigms.
     (Done. Described in Sections 1.1.1 and 1.1.2. Some related work demonstrated as early as SC06 and described in 2008 report.)
   - Demonstrate vgES with resource selection, resource binding, VG scheduling, for application kernels across large-scale grid platforms with diverse resource management paradigms.
     (Done. Described in Sections 1.1, 1.2 and 3.1.)
   - Validate and assess the integrated resource provisioning policies.
     (Done. Described in Section 1.)

ii. Execution System/Grid Economy:
   - Investigate adaptive pricing algorithms for resource reservations that accurately reflect their value when compared with typical best-effort queueing service.
     (Done. Described in 2008 report and Section 2.4)
   - Design experiment to investigate allocation efficiency under various pricing schemes.
     (Done. Described in 2007 and 2008 reports and Section 2.4.)
   - Target second VGrADS-enabled application (to be determined) as a driving application.
     (Done. Described in 2007 report.)
   - Verify using both GridSAT and second application.
     (Done. GridSAT and LEAD described in 2007 report.)

iii. Execution System/Fault Tolerance:
   - Integrate fault tolerance features into vgES and test on LEAD application.
     (Done. Described in Sections 1.3 and 3.1.)

iv. Programming Tools/Workflow:

- Incorporate novel techniques from vgES and fault tolerance work into workflow schedulers.
  (Done. Described in Sections 1.3 and 2.1 and 2008 report.)
- Study new problems in workflow based on grid economies, fault tolerance, and dynamic resource behavior.
  (*Done. Grid economies discussed in 2008 report. Scheduling for fault tolerance described in part in Sections 1.3 and 2.1. Dynamic resource behavior is a significant part of the cloud computing work described in Section 1.2, as well as the work in 2.2.)*
- Consider compilation approaches to task migration for fault tolerance.
  (Done. FTR work described in Sections 1.3.1 and 1.3.2, while not compiler-based, is relevant here. Also, static scheduling described in Section 2.1 relies on static information.)

v. Applications:
- Evaluate methods for scheduling workflow applications on large-scale TIGRE grid.
  (Partially done. TIGRE grid project diverged from VGrADS infrastructure, as discussed in 2007 report. TIGRE project completed in 2008)
- Explore additional TIGRE applications.
  (Not done. TIGRE grid project diverged from VGrADS infrastructure, as discussed in 2007 report. TIGRE project completed in 2008.)

vi. Education, Outreach, and Training:
- Continue AGEP program.
  (Done. Two students (Stephanie Diehl and Keisha Cumber attended AGEP summer program.)
- Continue graduate student exchanges.
  (Partially done. No further long-term exchanges were needed to complete projects, in the view of the PIs.)

5.2 Year 6 Milestones

As part of our requested No Cost Extension, we developed a set of milestones for finishing the project. Most of these are extensions of Year 5 milestones, specialized based on our current development. We made excellent progress on those milestones, as can be seen below:

i. Execution System/Virtualization:
- Extend vgES to manage dynamic “cloud computing” resources.
  (Done. Described in Sections 1.2 and 3.1)
- Investigate scheduling methods for systems with both allocated clusters and cloud computing resources.
  (Done. Described in Sections 1.2 and 3.1)

ii. Execution System/Grid Economy:

- Continue investigation of grid economy systems for resource allocation and scheduling.
  (Done. Described in 2008 report and Section 2.4.)

iii. Execution System/Fault Tolerance:
- Continue investigation of fault tolerance measures for cloud computing.
  (Done. Described in Section 1.3.)

iv. Programming Tools/Workflow:
- Demonstrate workflow schedulers sensitive to fault tolerance and performance model considerations.
  (Done. Described in Sections 2.1and 2.2.)

v. Applications:
- No additional milestones expected.

vi. Education, Outreach, and Training:
- Support 2009 Richard Tapia Celebration of Diversity in Computing.
  (Done. VGrADS PI Koelbel was on Tapia2009 committee, and we supported one student attendee (Keisha Cumber).)

6 Graduate Student Thesis Abstracts

During its six-year existence, VGrADS graduated eleven Ph.D. students and one M.S. student from seven different schools. Of these, two (Buneci and Ramakrishnan) were females, which roughly mirrors the 19% of doctorates awarded to women in recent years.

In response to NSF reporting requirements, we present here the thesis abstracts for our students. Full copies of most of their dissertations are available in the publications section of the VGrADS web site, http://vgrads.rice.edu/publications/.

6.1 Ayaz Ali

(Ph.D., 2008, University of Houston)
Adaptive Dynamic Scheduling of FFT on Hierarchical Memory and Multi-core Architectures

In this dissertation, we present a framework for expressing, evaluating and executing dynamic schedules for FFT computation on hierarchical and shared memory multiprocessor / multi-core architectures. The framework employs a two layered optimization methodology to adapt the FFT computation to a given architecture and dataset. At installation time, the code generator adapts to the microprocessor architecture by generating highly optimized, arbitrary size micro-kernels using dynamic compilation with feedback. At run-time, the micro-kernels are assembled in a DAG-like schedule to adapt the computation of large size FFT problems to the memory system and the number of processors.

To deliver performance portability across different architectures, we have implemented a concise language that provides specifications for dynamic construction of FFT schedules. The context free grammar (CFG) rules of the language are implemented in various optimized driver routines that compute parts of the whole transform. By exploring the CFG rules, we were able to dynamically construct many of the already known FFT algorithms without explicitly implementing and optimizing them. To automate the construction of best schedule for computing an FFT on a given platform, the framework provides multiple low cost run-time search schemes. Our results indicate that the cost of search can be reduced drastically through accurate prediction and estimation models.

With its implementation in the UHFFT, this dissertation provides a complete methodology for the development of domain specific and portable libraries. To validate our methodology, we compare the performance of the UHFFT with FFTW and Intel's MKL on recent architectures - Itanium 2, Xeon Clovertown and a second generation Opteron. Our optimized implementations of various driver routines compare favorably

against the FFTW and MKL libraries. Our experiments show that the UHFFT outperforms FFTW and MKL on most architectures for problems too large to fit in cache. Moreover, our low-overhead multithreaded driver routines deliver better performance on multi-core architectures.

6.2 Emma Buneci

(Ph.D., 2008, Duke University)
Qualitative Performance Analysis for Large-scale Scientific Workflows

Today, large-scale scientific applications are both data driven and distributed. To support the scale and inherent distribution of these applications, significant heterogeneous and geographically distributed resources are required over long periods of time to ensure adequate performance. Furthermore, the behavior of these applications depends on a large number of factors related to the application, the system software, the underlying hardware, and other running applications, as well as potential interactions among these factors.

Most Grid application users are primarily concerned with obtaining the result of the application as fast as possible, without worrying about the details involved in monitoring and understanding factors affecting application performance. In this work, we aim to provide the application users with a simple and intuitive performance evaluation mechanism during the execution time of their long-running Grid applications or workflows. Our performance evaluation mechanism provides a qualitative and periodic assessment of the application's behavior by informing the user whether the application's performance is expected or unexpected. Furthermore, it can help improve overall application performance by informing and guiding fault-tolerance services when the application exhibits persistent unexpected performance behaviors.

This thesis addresses the hypotheses that in order to qualitatively assess application behavioral states in long-running scientific Grid applications: (1) it is necessary to extract temporal information in performance time series data, and that (2) it is sufficient to extract variance and pattern as specific examples of temporal information. Evidence supporting these hypotheses can lead to the ability to qualitatively assess the overall behavior of the application and, if needed, to offer a most likely diagnostic of the underlying problem. To test the stated hypotheses, we develop and evaluate a general qualitative performance analysis framework that incorporates (a) techniques from time series analysis and machine learning to extract and learn from data, structural and temporal features associated with application performance in order to reach a qualitative interpretation of the application's behavior, and (b) mechanisms and policies to reason over time and across the distributed resource space about the behavior of the application.

Experiments with two scientific applications from meteorology and astronomy comparing signatures generated from instantaneous values of performance data versus

those generated from temporal characteristics support the former hypothesis that temporal information is necessary to extract from performance time series data to be able to accurately interpret the behavior of these applications. Furthermore, temporal signatures incorporating variance and pattern information generated for these applications reveal signatures that have distinct characteristics during well-performing versus poor-performing executions. This leads to the framework's accurate classification of instances of similar behaviors, which represents supporting evidence for the latter hypothesis. The proposed framework's ability to generate a qualitative assessment of performance behavior for scientific applications using temporal information present in performance time series data represents a step towards simplifying and improving the quality of service for Grid applications.

6.3 Walid Chrabakh

(Ph.D., 2006, University of California at Santa Barbara).
GridSAT: A Distributed Large Scale Satisfiability Solver for the Computational Grid

Grid Computing is an emerging field in computer science. Research in this area aims at aggregating distributed, heterogeneous and federated resources and make it available to Grid applications. In the past two types of applications have been deployed with varying degrees of success. The first type of applications is embarrassingly parallel (a bag of independent tasks). This category adapts well to a computational grid environment. The second category of applications includes mainly scientific code which is tightly coupled in nature. This type of applications is very hard to deploy in a grid environment.

In this thesis we present GridSAT, a new grid application. GridSAT is a distributed complete Boolean satisfiability solver based on the sequential solver Chaff. In addition to its theoretical significance, the satisfiability problem has numerous practical applications. SAT solvers are used in many engineering and scientific fields including circuit design and model checking.

GridSAT is able to achieve new results by solving faster those problems that were previously solved by other solvers. Moreover, it was able to solve problems which were left unsolved by other solvers. GridSAT accomplishes these results by achieving two goals. The first is parallelizing the sequential solver in a manner which allows it to run efficiently on a large collection of resources. GridSAT also uses techniques to enable information sharing between the parallel components to avoid redundant work. The second goal is to design and implement the application so that it can adapt to the dynamic conditions of a computational grid environment. The techniques and design used to realize GridSAT can be deployed with other application to achieve new results.

In addition, we show how multiple GridSAT instances can cooperate to run efficiently on a common set of resources without explicit synchronization. These experiments represent realistic scenarios where many grid applications share a common resource pool.

We have also developed a web portal which accepts problem instances through a standard web browser and returns status and results while shielding users from complexities of running the application manually.

6.4 Anshu Dasgupta

(Ph.D., 2006, Rice University).
Tailoring Traditional Optimizations for Runtime Compilation

Runtime compilation, due to its online nature, presents unique challenges and opportunities to compiler designers. Since compilation occurs during program execution, a just-in-time compiler (JIT) must be judicious in expending compilation time. The literature on traditional, offline compilers describes numerous program transformation techniques that strive to increase execution efficiency. However, while optimization passes for static compilers are well understood and have been thoroughly investigated, many such transformation algorithms cannot be implemented on a JIT environment due to compilation-time constraints. Further, offline algorithms are not designed to exploit information available to an online compiler at program execution time. The thesis of the research presented in this document is that program optimization techniques designed for traditional, offline compilers can be profitably adapted for a runtime compiler by effectively respecting the constraints imposed on compilation time and by exploiting the opportunities available in a runtime compilation environment. To that end, the dissertation explores the complexity of implementing program transformations for a runtime compiler and redesigns two optimization techniques for a JIT: register allocation and loop unrolling. The two transformations present contrasting challenges when they are included in a runtime compiler. While several offline, heuristic allocation algorithms achieve impressive results, they consume large amounts of compilation-time that are typically unacceptable for a JIT. We describe the design of two allocation algorithms that reduce allocation time while preserving the advantages of strong techniques authored for offline compilers. An experimental evaluation of the new algorithms demonstrates their effectiveness on a runtime compilation environment. While a runtime compiler is limited by the constraints imposed by its environment, compiling just prior to program invocation provides certain advantages over an offline compiler. In particular, it can examine information only available at program execution time. We describe the design of a lightweight runtime value-examining mechanism and a loop unrolling algorithm that work in tandem. Our experimental results indicate that the runtime unroller achieves significant improvements on floating point, scientific benchmarks. In summary, thus, the research described in this dissertation demonstrates how compiler optimization algorithms can be effectively tailored for runtime compilation.

6.5 Richard Huang

(M.S., 2005, University of California, San Diego).

Scheduling Compute Intensive Applications in Volatile, Shared Resource (Grid) Environments

While Grid computing has become popular in recent years, understanding Grid application performance remains a challenge. In open, shared resource Grid environments, applications face heterogeneity in resources and dynamic load on the resources. Variance in runtime prediction models leads to further variability in application performance. We investigate factors affecting application performance by extensive simulation using actual compute intensive applications runtimes and actual dynamic resource load. Our main result is that freshness of dynamic resource data does not matter for any data fresher than 300 seconds. Application performance varies less than 2% within this range. When a single application is scheduled on a volatile environment, scheduling optimistically improves performance as the scheduler can better tolerate volatility both in task runtime predictions as well as in the resource availability. Performance can be further improved by rescheduling tasks running much longer than predicted. Tasks running longer than predicted can indicate that the resource is overloaded or unavailable. When the number of resources equals or exceeds the number of tasks, the handling of tasks that have greatly exceeded their predicted runtime is critical for application performance. In some cases, rescheduling such tasks can improve application performance by as much as 2 to 3 times. In contrast, when there are more tasks than resources and multiple schedulers are competing for a set of resources, the opposite policy of allowing a late task to run to completion is preferred. This is because there are better resources for which to run that task and therefore rescheduling is unproductive.

6.6 Xin Liu

(Ph.D., 2004, University of California, San Diego).
Scalable Online Simulation for Modeling Grid Dynamics

Large-scale grids and other federations of distributed resources that aggregate and share resources over wide-area networks present major new challenges because they couple the behavior of resources and networks. These infrastructures support a new breed of applications which interact dynamically with their resource environment, making it critical to understand dynamic application and resource behavior to design for performance, stability, and reliability. Coupled use means that accurate study of dynamic applications, middleware, resource, and network behavior depends on coordinated, accurate, and simultaneous simulation of all four of these elements. Thus, the long-term challenge is to support scalable, high-fidelity, online simulation of applications, middleware, resources, and networks to support enable scientific and systematic study of grid applications and environments. That challenge is the focus of this dissertation. We define the problems in performing large-scale, high-fidelity, online simulation. We consider a number of approaches, and then present our approach in detail. Our approach includes a set of techniques which enable the use of real application and middleware software, and modeling of essentially arbitrary network and

resource properties. These techniques include resource virtualization via application interception, computation resource simulation based on soft real-time scheduling, and packet-level online network simulation. Our studies and experiments show that these techniques can support simulation experiments with complex software packages as well as resource and network structures. While most of the techniques in our approach are inherently scalable, one major challenge is online network simulation – which we implement as a parallel distributed discrete-event simulation, well-known to be challenging to scale. A range of techniques for scaling our online network are studied. Exploiting advanced graph partitioners, we explore a range of edge and node weighting schemes based on a variety of static network and dynamic application information. While simple approaches do not achieve acceptable load balance, our studies show that detailed network structure and behavior can be combined with the graph partitioners to achieve both good load balance and parallel efficiency. For example, our improvements increase efficiency and scalability by over 100 times, achieving a parallel efficiency of over 40\\% on 90-node clusters for a range of experiments. Our online simulation techniques are embedded in a working simulation tool, the MicroGrid, which enables accurate and comprehensive study of the dynamic interaction of applications, middleware, resource, and networks. We present experimental results with applications which validate the implementation of the MicroGrid, showing that it not only runs real grid applications and middleware, but also accurately models underlying resource and network behavior. Our scalability experiments show that our load balance algorithms are effective, and the best of them, hierarchical profile-driven load balance, scales well, enabling simulation networks of 20,000 routers with 90 cluster nodes. This is the largest detailed network simulation ever performed, and corresponds in size to a large ISP's network. Realistic packet level network simulation with tens of thousands of routers enables accurate study of grid and network dynamics at unprecedented scale, and we believe great opportunities for new insights.

6.7 Anirban Mandal

(Ph.D., 2006, Rice University).
Toward a Tool for Scheduling Application Workflows onto Distributed Grid Systems

In this dissertation, we present a design and implementation of a tool for automatic mapping and scheduling of large scientific application workflows onto distributed, heterogeneous Grid environments. The thesis of this work is that plan-ahead, application-independent scheduling of workflow applications based on performance models can reduce the turnaround time for Grid execution of the application, reducing burden of Grid application development. We applied the scheduling strategies successfully to Grid applications from the domains of bio-imaging and astronomy and demonstrated the effectiveness and efficiency of the scheduling approaches. We also proposed and evaluated a novel scheduling heuristic based on a middle-out traversal of the application workflow. A study showed that jobs have to wait in batch queues for a considerable amount of time before they begin execution. Schedulers must consider batch queue waiting times when scheduling Grid applications onto resources with batch

queue front ends. Hence, we developed a smart scheduler that considers estimates of batch queue wait times when it constructs schedules for Grid applications. We compared the proposed scheduling techniques with existing dynamic scheduling strategies. An experimental evaluation of this scheduler on data-intensive workflows shows that its approach of planning schedules in advance improves over previous online scheduling approaches. We studied the scalability of the proposed scheduling approaches. To deal with the scale of future Grids consisting of hundreds of thousands of resources, we designed and implemented a novel cluster-level scheduling algorithm, which scales linearly on the number of abstract resource classes. An experimental evaluation using workflows from two applications shows that the cluster-level scheduler achieves good scalability without sacrificing the quality of schedule.

6.8 Andrew Mutz

(Ph.D., 2008, University of California Santa Barbara).
Eliciting Honest Behavior on Computational Grids

During the last decade, a computing paradigm known as "grid" computing has seen a surge in research activity. Drawing inspiration from the electrical power grid, this paradigm is an approach to high-performance computing that seeks to serve a large number of users from multiple, geographically distinct computing centers. As grids exist today, users are competing for overcommitted resources. Without a well-designed mechanism to mediate the competing interests of users, the outcome can be chaotic and inefficient. Additionally, scheduling decisions are made based on user-submitted metadata, data that can be manipulated to increase a user's share of resources.

This dissertation explores the efficacy of auction-based schedulers as a means for mediating these competing interests. In particular, the use of auctions to incentivize honest disclosure of job metadata is investigated. We investigate this problem in the context of best-effort batch queues and reservation systems.

6.9 Daniel Nurmi

(Ph.D., 2008, University of California Santa Barbara).
Statistical Methods for Mitigating Resource Provisioning Dynamism in Large-scale Batch-scheduled Systems

Users of high performance computing (HPC) systems generally rely on concurrency to achieve performance. Modern users have the ability to draw from a vast array of distributed resources due to the ever-increasing quality of connecting software and networks. However, as the pool of resources available to users grows, so does the level of resource heterogeneity and performance response dynamism.

Historically, users request access to a super-computer's resources by submitting their work and waiting until the system has enough free resources to satisfy the user's request. However, few facilities exist that cater to the substantial class of users who

require that their work is completed by a specific time, who require that their resources are available during a specific time interval, or who require simultaneous access to multiple systems.

In this dissertation, we discuss new statistical methodologies to manage resource performance dynamism, and abstractions that build upon these methodologies to hide resource heterogeneity. In particular, we will show how we have successfully developed the methodologies and abstractions necessary to manage and hide provisioning delay of HPC resources.

6.10 Lavanya Ramakrishan

(Ph.D., 2009, Indiana University).
Multi-Level Adaptation for Performability in Dynamic Web Service Workflows

Large-scale computations from various scientific endeavors are composed as workflows that access shared data and high performance systems. Similarly, business applications in cloud computing systems use distributed infrastructure as part of mainstream business models. Recent advances in grid and cloud computing provide tools to monitor and manage execution. However, they do not provide predictable bounds on the Quality of Service (QoS) that can be expected in such variable multi-user distributed environments. Understanding the dynamic properties of resources and coordinated control of resources and workflows is critical especially for deadline-sensitive workflows such as weather prediction.

In this dissertation we revisit the software stack that supports the multi-tier services and propose and evaluate the WORDS (Workflow ORchestrator for Distributed Systems) architecture that abstracts the differences between specific resource models and provides a clear separation of concerns between the resource-level and application-level tools. In the context of the WORDS architecture we explore interfaces and mechanisms necessary for providing predictable quality of service to web service workflows with time and accuracy constraints.

We make the following four primary contributions. First, we propose a resource abstraction across grid and cloud resource control mechanisms that enables higher-levels tools to abstract the differences between systems. Second, we propose a probabilistic Quality of Service (QoS) model that enables providers to quantify the variation in resource availability; both for resource procurement due to competition and for the duration of the resource request from failures at various levels. Third, we use performability analysis through a Markov Reward Model to quantify the loss in performance and study the impact on cost due to availability variations. Finally, we propose a multi-phase orchestration approach that balances performance, reliability and cost considerations for a set of workflows.

6.11 Zhiao Shi

(Ph.D., 2006, University of Tennessee, Knoxville).
Scheduling tasks with precedence constraints on heterogeneous distributed computing systems

Efficient scheduling is essential to exploit the tremendous potential of high performance computing systems. Scheduling tasks with precedence constraints is a well-studied problem and a number of heuristics have been proposed.

In this thesis, we first consider the problem of scheduling task graphs in heterogeneous distributed computing systems (HDCS) where the processors have different capabilities. A novel, list scheduling-based algorithm to deal with this particular situation is proposed. The algorithm takes into account the resource scarcity when assigning the task node weights. It incorporates the average communication cost between the scheduling node and its node when computing the Earliest Finish Time (EFT). Comparison studies show that our algorithm performs better than related work overall.

We next address the problem of scheduling task graphs to both minimize the makespan and maximize the robustness in HDCS. These two objectives are conflicting and an epsilon-constraint method is employed to solve the bi-objective optimization problem. We give two definitions of robustness based on tardiness and miss rate. We also prove that slack is an effective metric to be used to adjust the robustness. The overall performance of a schedule must consider both the makespan and robustness. Experiments are carried out to validate the performance of the proposed algorithm.

The uncertainty nature of the task execution times and data transfer rates is usually neglected by traditional scheduling heuristics. We model those performance characteristics of the system as random variables. A stochastic scheduling problem is formulated to minimize the expected makespan and maximize the robustness. We propose a genetic algorithm based approach to tackle this problem. Experiment results show that our heuristic generates schedules with smaller makespan and higher robustness compared with other deterministic approaches.

6.12 Yang (Ryan) Zhang

(Ph.D., 2009, Rice University).
Grid-Centric Scheduling Strategies for Workflow Applications

Grid computing faces a great challenge because the resources are not localized, but distributed, heterogeneous and dynamic. Thus, it is essential to provide a set of programming tools that execute an application on the Grid resources with as little input from the user as possible. The thesis of this work is that Grid-centric scheduling techniques of workflow applications can provide good usability of the Grid environment by reliably executing the application on a large scale distributed system

with good performance. We support our thesis with new and effective approaches in the following five aspects.

First, we modeled the performance of the existing scheduling approaches in a multi-cluster Grid environment. We implemented several widely-used scheduling algorithms and identified the best candidate. The study further introduced a new measurement, based on our experiments, which can improve the schedule quality of some scheduling algorithms as much as 20 fold in a multi-cluster Grid environment.

Second, we studied the scalability of the existing Grid scheduling algorithms. To deal with Grid systems consisting of hundreds of thousands of resources, we designed and implemented a novel approach that performs explicit resource selection decoupled from scheduling. Our experimental evaluation confirmed that our decoupled approach can be scalable in such an environment without sacrificing the quality of the schedule by more than 10%.

Third, we proposed solutions to address the dynamic nature of Grid computing with a new cluster-based hybrid scheduling mechanism. Our experimental results collected from real executions on production clusters demonstrated that this approach produces programs running 30% to 100% faster than the other scheduling approaches we implemented on both reserved and shared resources.

Fourth, we improved the reliability of Grid computing by incorporating fault tolerance and recovery mechanisms into the workflow application execution. Our experiments on a simulated multi-cluster Grid environment demonstrated the effectiveness of our approach and also characterized the three-way trade-off between reliability, performance and resource usage when executing a workflow application.

Finally, we improved the large batch-queue wait time often found in production Grid clusters. We developed a novel approach to partition the workflow application and submit them judiciously to achieve less total batch-queue wait time. The experimental results derived from production site batch queue logs show that our approach can reduce total wait time by as much as 70%.

Our approaches combined can greatly improve the usability of Grid computing while increasing the performance of workflow applications on a multi-cluster Grid environment.

II. Findings

During VGrADS Year 6, VGrADS research continued to focus on three inter-institutional efforts: *VGrADS Execution System, VGrADS Programming Tools,* and *Applications*. The following sections summarize the findings of each subproject.

1 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.

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

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

Virtual Resource Reservations: We find that VARQ is able to "manufacture" a probabilistic resource reservation on systems that only support best-effort batch queue service. By predicting when a job should be submitted in the future in order to meet a specified deadline, the system ensures that the user will be guaranteed the resources during the desired timeframe (i.e. has a reservation). We find that VARQ reservations are often a preferable substitute for "hard" advanced reservations since the latter must currently be negotiated manually by members of our research team and the site administrators controlling the machines we currently target (e.g. the TeraGrid resources). Lastly, the Slotted Virtual Grid abstraction is essentially an abstract reservation that is translated directly (with little alteration) into a hard reservation once it is made by hand. We find that a VARQ reservation can support the SVG abstraction fully automatically with no intervention by the user or relevant system administrators, and can do so in production grid environments.

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

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

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

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

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. These results can be applied to either static (compile-time) or dynamic (run-time) scheduling.

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. In separate work, we showed that careful partitioning of a workflow into separate batch queue submissions could result in significant reductions in total (observed) wait time. Our scheduling algorithm based on this observation was remarkably robust under varying batch queue load policies, despite a lack of knowledge of those details.

Scheduling for Reliability: We have shown how applications can trade off reliability (probability of successful completion) and performance by a novel scheduling algorithm. The insight of the algorithm is that using resources with minimal {execution time}×{failure rate} maximizes reliability for any given makespan. Therefore, ordering processors by this quantity when choosing resources to use allows the application to optimize its reliability for a given deadline (or its execution time for a given reliability). This method can be applied to a variety of schedulers.

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

Fault-tolerance: From our experiments with various parameters for fault tolerance on slots, we have inferred that increasing reliability increments increases the median replication factor and the number of cases of replication failures, implying that replication techniques would be effective for moderate reliability increments. We can also infer that higher reliability of slots results in smaller median replication factors and fewer replication failures. We can also infer that lower reliability increments and higher slot reliabilities are desirable in order to increase the replication success rate.

Combined fault-tolerance and scheduling: From our experiments with combined fault-tolerance and scheduling on a set of workflows, we can infer that the over-provisioning mechanism can increase the workflow success probability by around 25% while lightweight checkpointing-restart can increase the success probability by around 12%. We have also shown that, using the combined techniques, we can obtain increased reliability with insignificant (5-6%) performance penalties.

3 Applications

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

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

GridSolve: We demonstrated a user-friendly interface to software running on distributed grid resources. Experimental results show that workflow computation is suitable to a certain subset of large grained tasks and can provide a performance gain in that situation.

Linear Algebra: Dynamic Task DAGs for Large-Scale Linear Algebra: Task-based distributed linear algebra algorithms were shown to be competitive with other approaches, and were proved to be scalable. In some algorithms, higher communication overheads reduce performance, but this is expected to have a lower impact as the number of cores per node increases.

III. VGrADS Education, Outreach, and Training Activities

The following sections describe VGrADS Education, Outreach, and Training (EOT) activities during VGrADS Year 6.

1 Training and Development Activities

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

1.1 Inter-institutional Collaboration

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

An important part of this interaction was student attendance at small-group workshops, which we call developer workshops. The demonstrations (at SC06, SC07, and SC08) of end-to-end VGrADS capabilities led us to emphasize these meetings rather than student exchanges as we did in early years of the project. The developer workshops are working meetings, producing detailed plans and software artifacts for use in our experiments and demonstrations. This year, we held one such meeting at ISI on September 8-9, 2008 to integrate cloud computing, virtual grid development, fault tolerance, and scheduling on the LEAD application. The students involved received significant experience in collaborative work and distributed software development, as well as a broader exposure to the project than they would ordinarily have had. Of course, the developer workshop also had a great positive effect on the demonstrations described elsewhere in this report.

1.2 Distributed Software Engineering

The VGrADS project has provided students with a chance to build a very large-scale system in which all of the components must work together efficiently. The students have learned goal-setting and management techniques for distributed teams, and have learned how to use a variety of group communication techniques to make distributed teams effective. Since research groups are developing components of the system at

various VGrADS sites, the project has also provided an opportunity for participants to collaborate closely with researchers with different expertise.

1.3 Courses

With support from their institutions, VGrADS PIs have developed and taught a variety of courses that cover Grid technologies and other aspects of high performance parallel and distributed computing. In the Spring 2009 semester, PI Jack Dongarra taught *CS 594 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. The same course had been taught in earlier years as well. For more information on this course and its contents, see the course web page at http://www.cs.utk.edu/~dongarra/WEB-PAGES/cs594-2009.htm. We refer the reader to previous annual reports for descriptions of other courses taught at UCSB (2005), UCSD (2005), UTK (2006, 2007 and 2008), and UH (2007 and 2008).

1.4 Students Produced

One of the great successes of VGrADS has been its production of graduate students. During the six years of the project, we have graduated 12 students, including two women. Those students are:

- Ayaz Ali (Ph.D., 2008, University of Houston).
  Thesis title: "Adaptive Dynamic Scheduling of FFT on Hierarchical Memory and Multi-core Architectures"
- Emma Buneci (Ph.D., 2008, Duke University).
  Thesis title: "Qualitative Performance Analysis for Large-scale Scientific Workflows"
- Walid Chrabakh (Ph.D., 2006, University of California at Santa Barbara).
  Thesis title: "GridSAT: A Distributed Large Scale Satisfiability Solver for the Computational Grid"
- Anshu Dasgupta (Ph.D., 2006, Rice University).
  Thesis title: "Tailoring Traditional Optimizations for Runtime Compilation"
- Richard Huang (M.S., 2005, University of California, San Diego).
  Thesis title: "Scheduling Compute Intensive Applications in Volatile, Shared Resource (Grid) Environments"
- Xin Liu (Ph.D., 2004, University of California, San Diego).
  Thesis title: "Scalable Online Simulation for Modeling Grid Dynamics"
- Anirban Mandal (Ph.D., 2006, Rice University).
  Thesis title: "Toward a Tool for Scheduling Application Workflows onto Distributed Grid Systems"
- Andrew Mutz (Ph.D., 2008, University of California Santa Barbara).
  Thesis title: "Eliciting Honest Behavior on Computational Grids"

- Daniel Nurmi (Ph.D., 2008, University of California Santa Barbara).
  Thesis title: "Statistical Methods for Mitigating Resource Provisioning Dynamism in Large-scale Batch-scheduled Systems"
- Lavanya Ramakrishan (Ph.D., 2009, Indiana University).
  Thesis title: "Multi-Level Adaptation for Performability in Dynamic Web Service Workflows"
- Zhiao Shi (Ph.D., 2006, University of Tennessee, Knoxville).
  Thesis title: "Scheduling tasks with precedence constraints on heterogeneous distributed computing systems"
- Yang (Ryan) Zhang (Ph.D., 2009, Rice University).
  Thesis title: "Grid-Centric Scheduling Strategies for Workflow Applications"

The abstracts of their theses are included as an appendix to the "Activities" section of this final report.

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.

We sponsored two female undergraduates during summer 2008. Out of many applicants, we selected Keisha Cumber (from Johnson C. Smith University in Charlotte, NC) and Stephanie Diehl (from Case Western University). Both are freshmen. They worked under the joint mentoring of Chuck Koelbel and Vivek Sarkar (a non-VGrADS-affiliated professor of CS) on projects related to parallel and distributed computing. After some tutorial material, they studied the mapping of task graphs representing scientific algorithms (Gaussian elimination and matrix multiply) onto multicore processors using the Habanero language developed by Sarkar. The project was successful, leading to a presentation by Cumber at the Tapia 2009 conference in

Portland, OR. Diehl returned to Rice for another AGEP experience in summer 2009, not directly funded by VGrADS.

In summer 2009, we sponsored one female student, Loren Micheloni. Under the direction of VGrADS co-PI Chuck Koelbel, Loren, now a senior in the computer science program at the University of Texas - Austin, implemented matrix multiplication on multicore processors using Habanero. Our original plans had been to use this routine as the basis for more interesting graph algorithms – specifically, Dijkstra's and Warshall's algorithms – but due to equipment failures we were unable to complete that more ambitious project. However, Loren certainly learned a lot during the summer and returned to Austin enthused about parallel computing.

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. Since our last annual report, the Tapia 2009 conference was held in Portland, OR on April 1-4, 2009. VGrADS PI Chuck Koelbel served on the program committee, co-chairing the posters competition (as he did in 2007). VGrADS also supported the travel of Keisha Cumber to present her summer project described in Section 2.1. We had offered to support Stephanie Diehl as well, but she was unable to attend due to course commitments.

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

VGrADS PIs have continued to participate in the NSF-funded Computer Science Computer and Mentoring Partnership (CS-CAMP) project (http://ceee.rice.edu/cs-camp/). VGrADS PIs Keith Cooper and Richard Tapia are PIs of the CS-CAMP project. MS CS-CAMP, an extension of the original CS-CAMP program for high school girls, is designed for middle school girls. With support from VGrADS, Tapia (with the help of Michael Sirois) was able to host the summer program in both 2007 and 2008. Both years attracted between 30 and 45 students.

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

VGrADS PIs Keith Cooper, Richard Tapia, and Chuck Koelbel, who were involved in planning the new program, participated in sessions during both years of the camp.

2.4 Open Education Cup Competition

As an experiment at SC08, Rice University (with sponsorship from several industrial partners) announced the Open Education Cup. This was a competition to create educational modules for use by everyone who wanted to teach and learn about High Performance Computing (HPC) and parallel computing. Recognizing that parallel and distributed computing play a large and growing role in science and engineering, and recognizing that study and education materials are often unavailable or expensive, we took the initiative to help create creating Open Educational Resources (OER) in the area of HPC.

The idea behind OER is to make high-quality educational material freely available to teachers and learners so that they can master and enhance the material, in the same way that the Open Source movement has furthered the production of software. Once OER content exists, it can be picked up, used, and improved by a large and growing community. This creates a vibrant "ecosystem" for teaching and learning about a topic. If the parallel computing community embraces OER and starts sharing its knowledge, it can rapidly build the workforce it needs, disseminate new information, and take the field to new heights. To encourage this, the Open Education Cup offered cash prizes and NVIDIA graphics cards as inducement to create such content. All submissions were published under the Creative Commons license (a form of open source copyright) in the Connexions project (http://www.cnx.org/) at Rice University, a technology platform supporting OER for authors, educators and learners.

VGrADS PI Chuck Koelbel, in collaboration with Jan Odegard (also at Rice) spearheaded the Open Education Cup, organizing and publicizing the competition. In particular, Koelbel penned the introductory module for the competition, available at http://cnx.org/content/m18099/latest/. Koelbel and Odegard served as judges for the submissions as well. Although the contest did not attract as many submissions as we had hoped, and therefore will not be repeated next year, it did attract substantial attention and encouraged a good deal of discussion of open education at the conference.

2.5 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 SC08 Conference (Austin, TX, November 15-21, 2008). Other sections of this report

detail the research advances that were reported there. VGrADS researchers also gave a number of talks with accompanying demonstrations in exhibit booths, including:

- Chuck Koelbel (Rice), Rich Wolski (UCSB), Anirban Mandal (RENCI), Dan Nurmi (UCSB) and Lavanya Ramakrishnan (IU) gave overviews of the VGrADS project in the GCAS booth (a collaboration of Rice, UH, and Texas A&M), SDSC booth, and RENCI booth.

- Lavanya Ramakrishnan (IU), Anirban Mandal (UNC), and Dan Nurmi (UCSB) led a team of students and staff in demonstrating the LEAD application running with VGrADS for both performance and fault tolerance.

- Andrew Mutz and Rich Wolski (both UCSB) presented the VGrADS-related paper "Efficient Auction-based Grid Reservations using Dynamic Programming"

- Emma Buneci (Duke) and Dan Reed (Microsoft) presented the paper "Analysis of Application Heartbeats: Learning Structural and Temporal Features in Time Series Data for Identification of Performance Problems" based on Buneci's thesis.

- Lamia Youseff (UCSB), a student of Rich Wolski, presented her work on "Paravirtualization Performance and Programming Support for Next Generation HPC System" at the Doctoral Showcase.

- Chuck Koelbel (Rice) helped organize and gave presentations about the "High Performance Computing Cup", a competition to create educational material about parallel computing.

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

IV. VGrADS Contributions

1. Contributions within Discipline

VGrADS activities and findings during Year 6, 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 include:

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

  In VGrADS Year 6, VGrADS researchers extended the implementation of virtual grids to include cloud computing instantiations. They showed that unified grids containing TeraGrid and EC2 resources, as well as local batch-queued and cloud resources, could be used to address deadline-driven scientific applications. They verified that the VGrADS abstractions were capable of implementing advanced applications.

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

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

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

estimations of application performance and batch queue wait times to generate high-quality schedules in slotted virtual grid environments. This year, we developed additional methods for executing workflow applications on batch queue controlled resources, and (in separate work) integrated overprovisioning and checkpointing into list-based schedulers.

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

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

- VGrADS researchers have studied the feasibility of traditional compiler optimizations for grid computing systems. Previous reports have documented the effectiveness of optimizations such as register allocation in the context of just-in-time compilation. This year, we have studied the requirements of system-specific auto-tuning of FFT software, which can be used to significantly improve individual node performance in a grid environment. This work supports the VGrADS philosophy of using local compilation to enable heterogeneous executables.

- VGrADS researchers have developed the FT-MPI library to provide process-level fault tolerance based on the MPI 1.2 standard, with excellent performance (comparable to MPICH2 or LAM). This work is being incorporated in the OpenMPI project, which is creating a completely new MPI-2 implementation using the best library technologies and resources available. New work this year has included evaluation of the Binomial Graph Network as a basis for fault-tolerant message passing layers.

2) Contributions to Other Disciplines

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

3) Contributions to Human Resources Development

The VGrADS project has provided computer science research opportunities for graduate students and postdoctoral associates, including individuals from underrepresented groups. Through participation in VGrADS project meetings, email, and phone conversations, students and postdoctoral associates have been able to interact with, learn from, and contribute toward the research of off-site VGrADS participants. The multi-site nature of this project has exposed participants first-hand to a wider range of research approaches and specialty areas than would typically be possible. Notably, this includes discussion and collaboration with several world-renowned researchers from other institutions with whom the students would not normally interact.

With support from their institutions, VGrADS PIs continue to develop and teach a variety of courses that cover Grid technologies and other aspects of high performance parallel and distributed computing. Most notably, this includes the Computational Science for Engineers course that VGrADS PI Jack Dongarra has regularly taught since 2007.

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

VGrADS researchers have presented VGrADS research results and ideas in a variety of forums, including technical computer science presentations, presentations to applications groups, and presentations to students. In particular, the VGrADS project was involved in a variety of outreach activities at the SC08 conference in Austin, TX (November 15-20, 2008). VGrADS activities at SC08 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.