# II. Findings

During the reporting period (6/1/02-5/31/03), GrADS research has focused on five inter-institutional efforts: *Program Execution System (PES), Program Preparation System (PPS) & Libraries, Applications, MacroGrid & Infrastructure,* and *MicroGrid*. The following sections summarize the findings of each subproject.

## 1 Program Execution System (PES)

The major findings of the GrADS efforts in the program execution system area are as follows:

- It is possible to construct reasonably efficient schemes for dynamic rescheduling of Grid applications onto different resources during execution if performance problems mandate it.
- Process swapping appears to be an effective means of improving performance for iterative applications on distributed systems with varying load, with performance comparable to checkpoint/restart and dynamic load balancing. A key to success is choosing an appropriate policy for choosing when to migrate a process. Furthermore, process swapping in MPI can be implemented in a way that is (mostly) transparent to the user application.
- Many existing migration frameworks that migrate applications under loading conditions implement simple policies that cannot be applied to Grid systems. We have implemented a migration framework that takes into account both the system load and application characteristics.
- Within the GrADS framework, we need scheduling mechanisms that take into account requirements for software installations at various Grid resources, to permit linking Grid applications against pre-installed libraries.
- We have demonstrated that our expanded contract monitoring infrastructure, working in concert with the GrADSoft toolkit, can be used to instantiate, evaluate and visualize performance contracts. Our tests have shown that the infrastructure can be applied to applications from different areas, and that it supports the monitoring of computation and communication metrics collected from execution sites widely distributed. Our extensive experiments with new approaches to application and system monitoring have proven that these methodologies are suitable for use in the monitoring of Grid applications comprising long-running executions on a large set of resources.
- We have demonstrated that an assembly-level approach to insertion of sensors and actuators for contract monitoring can be effective, although performance becomes an issue on large object files.
- A complete redesign and re-vamping of the Launcher is being planned for the coming year to provide enhanced capabilities for monitoring and operation, based on the performance studies undertaken during this reporting period with the applications of FASTA, UCSD rescheduler and CACTUS.

## 2 Program Preparation System (PPS) and Libraries

The major findings of the program preparation system effort are summarized as follows:

- To construct accurate, scalable, machine-independent models of application performance, detailed analysis of an application's instruction sequence is necessary, both to understand how computational kernels map to the resources of a target architecture and to understand the performance impact of exposed memory latency for a particular architecture and problem size.
- A combination of static analysis, automatically-inserted instrumentation, and analysis of data from moderate-sized benchmark runs allows the construction of portable black-box performance models. In particular, fitting of polynomial models to trace data seems to give good results for predicting computation and memory performance. We are continuing work to extend these results to more distributed programs.
- Automatic construction of mappers for programs written in MPI, HPF, or other parallel languages can be based on clustering applied to the familiar task graph abstraction, which is used as a basis several major performance-modeling projects. This allowed us to perform the first execution of an HPF program over the Grid.
- Our Vizer framework conducts high-level optimizations on binary programs. Vizer analyzes binary files and reconstructs data structures and control flow that were present in the high-level source code used to create the binary. This information can be used to instrument x86 executables on Grid computing environments and to implement optimizations that are otherwise not possible in binary optimizers such as the vectorization of Intel x86 object code.
- To build reliable and re-usable Grid applications, programmers must be equipped with a programming framework that hides the details of most Grid services and allows the developer a consistent, non-complex model in which applications can be composed from well tested, reliable sub-units. GrADS has investigated a software component framework for building Grid applications based on the DOE Common Component Architecture (CCA), which allows individual components to export function/service interfaces that can be remotely invoked by other components.
- The UHFFT library adapts automatically to the hardware it is running on by using a dynamic construction (execution phase) of composable blocks of code generated and optimized for the underlying architecture during the installation of the library (program preparation). Its performance is very competitive with the best-known public domain libraries and even some vendor libraries, and GrADS benchmarking efforts promise to eventually being able to model it in a form amenable for executable and composable performance models.
- The ScaLAPACK performance model library has shown initial promise and can be used even outside the GrADS project.

## 3 Applications

The major findings of the GrADS efforts related to GrADS applications are summarized as follows:

- High performance can be achieved on the Grid for several different kinds of numerical applications with a low implementation and execution overhead due to the GrADS framework.
- GrADS can handle applications that access large databases that should not be moved over a wide area network, scheduling the computation at the site of the data.
- GrADS can handle applications from varying disciplines with varying requirements, such as the master-worker sequence alignment application FASTA.

- GrADSAT showed that distributed and parallel methods can be applied to solve the propositional satisfiability problem, contrary to folklore in the field. Two important innovations - scalable distributed learning and adaptive resource scheduling –together result in an automatic SAT solver that outperforms the best previously known solvers.
- Preliminary work suggests that Grid computing may be a viable approach to the computationally demanding problem of determination of 3-D structure of large macromolecular complexes from electron cryomicroscopy.
- The NetSolve-GrADS proxy server enables the use of simple front ends (e.g., Matlab) to access the GrADSoft framework and libraries.
- GrADSolve is an RPC system for efficient execution of remote parallel software. The efficiency is achieved by dynamically choosing the machines used for parallel execution and staging the data to remote machines based on data distribution information. The GrADSolve RPC system also supports maintaining and utilizing execution traces for problem solving. Our experiments showed that the GrADSolve system is able to adapt to the problem sizes and the resource characteristics and yielded significant performance benefits with its data staging and execution trace mechanisms.

## 4 MacroGrid & Infrastructure

The MacroGrid has served to produce valuable Grid performance trace data reflecting resources usage patterns in Grid research settings. Because the testbed is used to run application-based verifications, the resource loads that it experiences exemplify "production" application runs in Grid settings.

Applications typically go through a debugging phase and then a series of full-scale tests are executed to generate results. We have been collecting resource performance data from the Macro Grid continuously since the first up-grade in November 2002. Analysis of this data has revealed new insights into the predictability of resource loads and their dynamics. Because the Macro Grid is a working Grid, but one that provides an advanced set of features, it allows us to anticipate the future impact of our results on production computing settings.

Both the performance data generated by the Macro Grid and several of the GrADSoft tools have been used as the basis of a project-oriented graduate course in Computational Grid computing. In it, students study the dynamics of performance fluctuations; design applications and schedulers based on this analysis, and implement their solutions using Grid tools. Thus, the GrADS software tools and the Macro Grid performance data have enabled new educational experiences for students.

Main findings of the MacroGrid during this past year included:

- Fulfilled a critical role in development and testing of GrADS software.
- Continued to provide a stable execution environment for testing GrADSoft components through the transition to updated versions of Grid software.
- Expanded the role of information services within the GrADS software framework.
- Introduced significant heterogeneity into the MacroGrid through the inclusion of UH's IA-64 cluster. The effort in bringing up the GrADSoft environment on the UH cluster did

not pose any difficulties with the GrADSoft environment other than the "normal" issues with the underlying middleware, firewalls, and the local scheduling environment at UH.

We have found that the role of information services can be extended to provide additional capabilities required within the GrADSoft framework. For an application to run within the GrADSoft framework, additional pieces of information will be required to be available. These items could include location of parameter files, location of executables required for staging prior to execution, location of standard input/output/errors, list of target resources for executing binaries etc. During the reporting period, we have extended the information services capabilities provided by MDS 2.2 and develop a GrADS Information Repository. This repository can be queried for appropriate information to make suitable decisions on resource selection and contract monitoring etc

## 5 MicroGrid

The major findings of the MicroGrid effort are summarized as follows:

- Understand and demonstrate that the MicroGrid can be used to study real application runs in complex Grid environment. This reflects successfully virtualizing the functional interfaces of resources and their performance capabilities. Experiments showed that insights into application middleware performance can be achieved under MicroGrid
- Demonstrated good performance for applications, enabling exploration with real applications and full data sets. For example, we have demonstrated application experiments at as much as 1/4 th speed.
- Clearly demonstrate that scalable network emulation is a challenging problem. Load imbalances are severe, and they provide a significant challenge to scalability. We have characterized the performance of basic topology-driven and simple topology-driven + applications, and they don't achieve sufficient load balance to scale beyond 10's of physical emulation nodes. Our initial profiling technique is significantly better, but needs to improve significantly to achieve scaling to hundreds of physical resource nodes.