

The MicroGrid: Using Online Simulation to Predict Application Performance in Diverse Grid Network Environments

Huaxia Xia^{*}, Holly Dail[†], Henri Casanova^{*†}, Andrew A. Chien^{*}

^{*}Department of Computer Science and Engineering
University of California at San Diego

[†]San Diego Supercomputer Center
University of California at San Diego

[hxia,hdail,casanova,achien]@cs.ucsd.edu

Abstract

Improvements in networking and middleware technology are enabling large-scale grids that aggregate resources over wide-area networks to support applications at unprecedented levels of scale and performance. Unfortunately, existing middleware and tools provide little information to users as to the suitability of a given grid topology for a specific grid application. Instead, users generally use ad-hoc performance models to evaluate mappings of their applications to resource and network topologies. Grid application behavior alone is complex, and adding resource and network behavior makes the situation even worse. As a result, users typically employ nearly blind experimentation to find good deployments of their applications in each new grid environment. Only through actual deployment and execution can a user discover if the mapping was a good one. Further, even after finding a good configuration, there is no basis to determine if a much better configuration has been missed. This approach slows effective grid application development and deployment.

We present a richer methodology for evaluating grid software and diverse grid environments based on the MicroGrid grid online simulator. With the MicroGrid, users, grid researchers, or grid operators can define and simulate arbitrary collections of resources and networks. This allows study of an existing grid testbed under controlled conditions

or even to study the efficacy of higher performance environments than are available today. Further, the MicroGrid supports direct execution of grid applications unchanged. These application can be written with MPI, C, C++, Perl, and/or Python and use the Globus middleware. This enables detailed and accurate study of application behavior. This paper presents: (1) the first validation of the MicroGrid for studying whole-program performance of MPI Grid applications and (2) a demonstration of the MicroGrid as a tool for predicting the performance of applications on a range of grid resources and novel network topologies.

1. Introduction

Rapid improvements in the performance of wide-area networks and the pervasive deployment of commodity resources provide us grid computing infrastructures with tremendous potential [4]. Today, this potential is being widely tapped, and many grid middleware projects such as Globus [16], Condor [29], and NetSolve [1] have been pursued to provide uniform, secure and efficient access to remote resources. Unfortunately, there are a paucity of tools that assist a user in predicting *if* their application will obtain suitable performance on a particular platform. Instead, with little information on likely performance, users must invest significant time to obtain accounts on the new grid, adapt their application to new middleware, debug their grid executions, and finally run experiments to determine the best deployment of their application on the new grid. At last the user knows if the blind date was a good one: did the application run efficiently on the grid? If not, it is time to set up a new blind date and start over. Two major problems with this ap-

This material is based upon work supported in part by the National Science Foundation under awards NSF EIA-9975020 Grads, NSF ACI-0103759, NSF Cooperative Agreement ANI-0225642 (OptiPuter), NSF CCR-0331645 (VGrADS), NSF ACI-0305390, and NSF Research Infrastructure Grant EIA-0303622. Support from Hewlett-Packard, BigBangwidth, Microsoft, and Intel is also gratefully acknowledged.

proach are that it is both labor and resource intensive, and does not provide any assurance of the quality of results. As resource environments and application performance structure continues to increase in complexity, it is likely the distance between achieved results and optimal will increase further. As resource environments become increasingly dynamic, it is likely such a blind exploration methodology will become even more unworkable.

The goal of the MicroGrid project is to develop and implement simulation tools that provide a vehicle for the convenient scientific study of grid topologies and application performance issues. The MicroGrid provides a virtual grid infrastructure that enables scientific experimentation with dynamic resource management techniques and adaptive applications by supporting controlled, repeatable experiments. The MicroGrid complements experimentation with actual grid testbeds because the MicroGrid can be used to explore a wide variety of grid resource configurations and scenarios (such as catastrophic failure), which may not be possible to exhibit in the actual resources. Further if application or middleware behavior is difficult to model accurately, the use of direct application execution enables accurate modeling. The Microgrid provides reduced setup effort for simulation and increases the observability of application behavior.

Previous papers have provided a broad overview of an early version of the MicroGrid [27], and studied approaches for load-balancing the workload of network simulation to enable scalable MicroGrid simulation [20]. This paper provides the following contributions.

- An extension of MicroGrid Toolkit features including added support for simulation of grid components written in C, C++, Python, and Perl.
- The first validation of the efficacy of the MicroGrid simulation approach for whole-program MPI applications. The paper presents experiments comparing the measured application behavior of five grid applications in real-world testbed environments against their behavior as measured in simulation. Note that our goal is not perfect prediction, but rather to provide users with some expectation of application performance in new topologies.
- The first demonstration of the MicroGrid as a tool for predicting the behavior of whole-program MPI applications on grid environments without the need for real-world access to those environments. These experiments explore topologies in existence today such as the TeraGrid [28] and high-performance network topologies that do not yet exist.

The remainder of the paper is organized as follows. Section 2 gives some background on the MicroGrid and Sec-

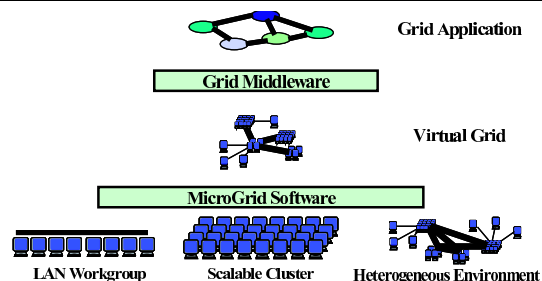


Figure 1. Architecture of the MicroGrid Online Simulation Toolkit.

tion 3 describes the five MPI applications we deployed in the real-world and the MicroGrid. Next, Section 4 presents experiments validating the MicroGrid and applying it to novel grid topologies. Section 5 provides an overview of related work and Section 6 concludes the paper and highlights future directions.

2. MicroGrid Architecture

We present a high-level overview of the MicroGrid (see [27, 20] for a more complete design discussion). The basic functionality of the MicroGrid allows grid experimenters to execute their applications in a virtual grid environment. The virtual environment presents the application a view of virtual grid information services and virtual resources. These virtual resources are of course run on actual physical resources, and the MicroGrid can exploit either homogeneous or heterogeneous resources (see Figure 1). The MicroGrid virtualizes two different grid resources types: network resources and compute resources. This virtualization is achieved by the MicroGrid network online simulator (MaSSF) and CPU controller, respectively. We describe each of these components below. Note that the MicroGrid ensures coherent virtual executions by coordinating the simulation speed of the different virtual resources.

2.1. MaSSF: Scalable Network Simulation

MaSSF (pronounced “massive”) is a scalable packet-level network online simulator that supports direct execution of unmodified applications. MaSSF consists of four parts.

- **Simulation Engine:** MaSSF uses a distributed simulation engine based on DaSSF [9, 19]. It utilizes MPI-connected cluster systems to achieve scalable performance. We also implement a real-time scheduler in order to enable best-effort simulation. This scheduler

can also run in a scaled-down mode when the simulated system is too large to be simulated in real time on available hardware. With the global coordination of the MicroGrid, this feature provides extreme flexibility to simulate a wide range of networks accurately.

- **Network Modeling:** MaSSF provides necessary protocol modules for detailed network modelling, such as IP, TCP/UDP, OSPF, and BGP4. We have strived to simplify these protocols and maintain their behavior characteristics at the same time. We also use a network configuration interface similar to a popular Java implementation, SSFNET [9], for user convenience.
- **Simulation Capability:** To support simulation of traffic from live applications, we implemented an Agent module to accept and dispatch live traffic from applications to the network modeling module. Traffic will also be sent back to application through the Agent Module.
- **Live traffic interception:** Application processes use a module called WrapSocket to intercept live network streams at the socket level. The WrapSocket then talks with the Agent module to redirect traffic into the network simulator and vice versa. WrapSocket can be either statically or dynamically linked to application processes and requires no application modification.

2.2. The MicroGrid CPU controller

The CPU controller virtualizes the CPU resources and processes of the physical machines by sending SIGSTOP and SIGCONT signals to processes. The controller consists of three parts:

- **Live Process Interception:** Whenever a process or a thread is created or is destroyed, the CPU controller detects the event via intercepted `main()` or `exit()` function calls and updates its internal process table.
- **CPU Usage Monitoring:** Every 20ms, the controller reads the `/proc` file system to check the CPU usage of all the processes in its process table.
- **Process Scheduling:** The controller calculates the CPU usage of each virtual host (in a sliding window). If the amount of effective cycles exceeds the speed of the virtual hosts, the controller sends a SIGSTOP signal to all processes of the virtual host; otherwise, the controller wakes up the processes and let them proceed. As in MaSSF, the CPU controller also supports scaled-down mode to simulate virtual machines which are faster than available physical resources.

This architecture allows simulation of large numbers of machines (100's to thousands) on a small number of ma-

chines. Further, heterogeneous performance from slow to fast machines can be modeled accurately.

3. Applications

In this section we describe five classic grid applications used for research and development in the GrADS project [5, 13]. We will use these applications for MicroGrid validation experiments in the following section.

All five applications are SPMD MPI applications and have been previously tested on the GrADS testbed in various real-world experiments. These applications were integrated into the GrADS framework and tested in various experiments as part of the following efforts: ScaLAPACK [23], Jacobi [10], Game of Life [10], Fish [26], and FASTA, which was integrated by Asim YarKhan.

ScaLAPACK [6] is a popular software package for parallel linear algebra, including the solution of linear systems based on LU and QR factorizations. We use the ScaLAPACK right-looking LU factorization code based on 1-D block cyclic data distribution. The application is implemented in Fortran with a C wrapper. The data-dependent and iteration-dependent computation and communication requirements of ScaLAPACK provides an important test for the MicroGrid online simulation. In our experiments we used a matrix size of 6000×6000 .

FASTA [22]. The search for similarity between protein or nucleic acid sequences is an important and common operation in bio-informatics. Sequence databases have grown immensely and continue to grow at a very fast rate; due to the magnitude of the problems, sequence comparison approaches must be optimized. FASTA is a sequence alignment technique that uses heuristics to provide faster search times than more exact approaches, which are based on dynamic programming techniques. Given the size of the databases, it is often undesirable to transport and replicate all databases at all compute sites in a distributed grid. We use an implementation of FASTA that uses remote, distributed databases that are partially replicated on some of the grid nodes. FASTA is structured as a master-worker and is implemented in C. For MicroGrid validation purposes, an important aspect of FASTA is that each processor is assigned a different database (or portion of a database) so the MicroGrid must properly handle input files and provide proper ordering of data assignments onto processors. In our experiments the sizes of the databases are 8.5MB, 1.7MB and 0.8MB respectively. The query sequence is 44KB.

Jacobi uses Jacobi iteration [3] to solve a linear system of equations solver. A portion of the unknown vector x is assigned to each processor. During each iteration, every processor computes new results for its portion of x and then broadcasts its updated portion of x to every other processor. Jacobi is a memory-intensive application with a com-

munication phase involving lots of small messages. In our experiments we used a matrix size of 9600×9600 .

Fish models the behavior and interactions of fish and is indicative of many particle physics applications. The application calculates Van der Waals forces between particles in a two-dimensional field. Each computing process is responsible for a number of particles that move about the field. The amount of computation depends on the location and proximity of particles, so Fish exhibits a dynamic amount of work per processor. In our experiments we used 6,000 particles.

Game of Life implements Conway's Game of Life [12]; a well-known binary cellular automaton. A two-dimensional mesh of pixels is used to represent an environment of cells. In each iteration every cell is updated with a 9-point stencil and then processors send data from their edges (ghost cells) to their neighbors in the mesh. Game of Life has significant memory requirements compared to its computation and communication needs. In our experiments we used a matrix size of 9600×9600 .

3.1. Summary

Our set of applications provides a varied test suite for the MicroGrid online simulator as it includes coverage of a wide range of application characteristics of interest.

- Fortran (ScaLAPACK) vs. C (the others)
- Floating point intensive (ScaLAPACK, Jacobi, and Fish) vs. integer intensive (Game of Life and FASTA)
- Varying, complicated sharing communication phases (ScaLAPACK) vs. regular, straightforward exchange of vector(s) all-to-all (Jacobi, Fish) vs. neighbor-to-neighbor communications only (Game of Life) vs. master-worker send-receive (FASTA)
- Full range of communication to computation balance from highly synchronized, large number of messages (ScaLAPACK) to small number of master-worker send-receive pairs (FASTA).

4. Experiments

In this section we describe experiments we performed using the MicroGrid and the five grid applications described in the previous section. In the first set of experiments, we test how accurately the MicroGrid simulates application execution behavior. For these experiments, we compare real-world executions of grid applications against MicroGrid simulations of the same environments. In the second set of experiments, we simulate a number of new, high-performance grid architectures and test the performance of our applications on these simulated architectures. These

tests demonstrate how the MicroGrid can be used to evaluate applications and architectures that are not available in the real-world (or where controlled experiments are not possible).

4.1. MicroGrid Validation

In our first set of experiments, we compare grid application behavior for real-world application executions against the behavior observed in the MicroGrid online simulation. Specifically, we selected a multi-site grid and single-site cluster from the GrADS project testbed and ran the five grid applications on these two testbeds. Next, we built a MicroGrid resource model of the grid and cluster including the speed, architecture, and number of compute nodes and the TCP/IP configuration, latency and bandwidth of the links. Then, we executed the same applications on the MicroGrid simulation.

4.1.1. Real-world Testbed Configuration A more detailed view of our testbed choices may be useful in considering which aspects of the MicroGrid online simulation have been exercised by the validation. The GrADS project has developed a testbed called the MacroGrid, used for, among other things, development and testing of new grid technologies. The MacroGrid currently includes over 100 machines (see Figure 2), of which the following 10 machines were used in our studies.

- **UCSD cluster:** three 2100+ XP Athlon AMD (1.73 GHz) with 512 MB RAM each. These systems run Debian Linux 3.0 and are connected by Fast Ethernet.
- **UIUC cluster:** three 450 MHz PII machines with 256MB memory connected via TCP/IP over 1Gbps Myrinet LAN. These systems run RedHat Linux 7.2.
- **UTK cluster:** four PIII 550 MHz machines with 512MB memory, running RedHat Linux 7.2, and connected with Fast Ethernet.

The three sites are connected by the Internet2 network with 2.4Gbps backbone links. During our experiments, we observed NWS latency and bandwidth values over a period of 12 hours and obtained ranges as shown in Figure 2. For local area networks the NWS measured bandwidth via 64 KB messages; for wide area networks the NWS used 1 MB messages.

For our experiments, we label our test cases **GrADS Grid**, including 3 machines from each of the 3 sites listed above, and a **GrADS Cluster**, including just the 4 machines from the UTK cluster. We selected a relatively small subset of the GrADS testbed for experimentation for two reasons: we needed consistent access to the testbed machines (quite difficult for larger testbed subsets), and we needed application executions that could be simulated in a timely manner

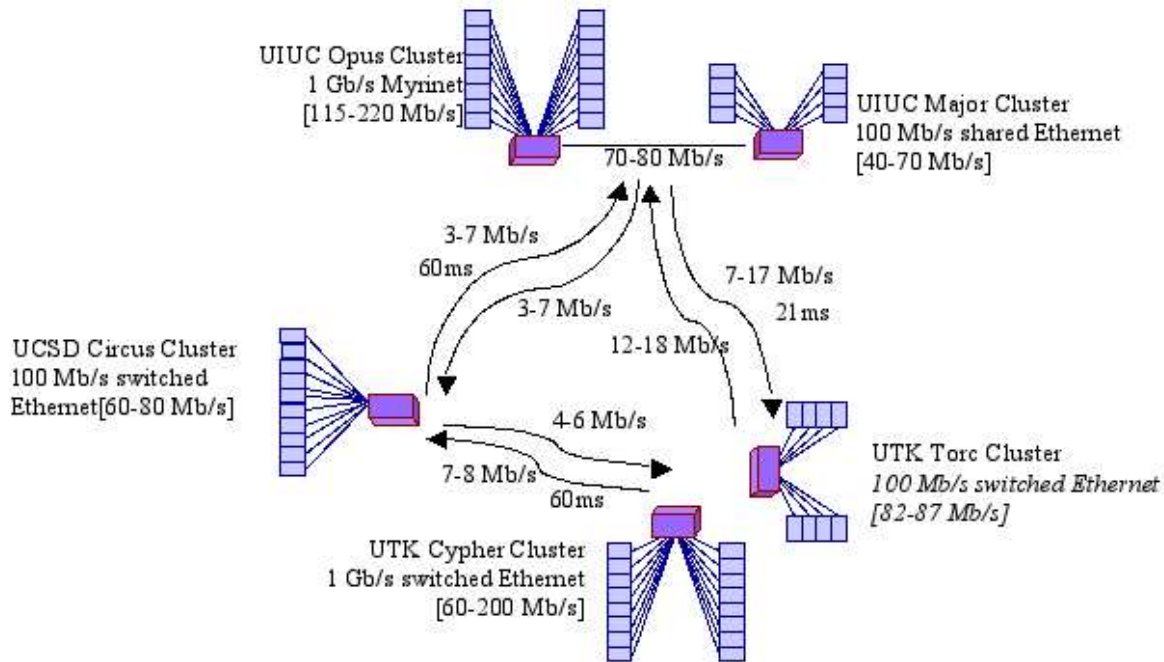


Figure 2. GrADS testbed with network links annotated with bandwidth and latency values collected with the NWS.

on the MicroGrid (simulation speed is partly dependent on the size of the testbed).

4.1.2. Microgrid Simulation Configuration Several steps are necessary to build an simulation environment in the MicroGrid. The first steps involve configuring the simulation environment for a particular testbed. We only had to perform this step once each for the GrADS Grid and GrADS Cluster; the simulation environments could then be used for each of the five applications with only a small change to processor speed assumptions, as explained in the next paragraph.

First, the set of machines to be included in the simulation has to be defined in a configuration file and each machine should be annotated with its processor speed and memory capacity. We wanted to simulate a variety of AMD and IA32 processors, while the MicroGrid compute platform contained only one type of IA32 processors. To obtain a mapping between processors, we first ran each of the five applications in single-process mode on one machine of each type to get an appropriate machine speed mapping.

Next, the network connecting the nodes must be defined and virtualized on top of an existing network. The MicroGrid virtual network topology is defined by latency and TCP sender window size. Based on NWS data collected from the GrADS testbed, we assume all the machines have

64KB TCP window size and we assume the following latencies between hosts: 31ms between UCSD and UIUC hosts; 30ms between UCSD and UTK hosts; and 11 ms between UIUC and UTK hosts.

We ran the applications on nine dual 450MHz PII machines and the MaSSF network simulator on two dual 2.4GHz Xeon machines. The simulation rate was five, i.e., the applications ran five times slower than real time.

At the time of our experiments the MicroGrid did not support injection of load traces; thus, for these initial experiments we sought relatively unloaded processors in the real-world and then modeled the environment as unloaded in the MicroGrid. Of course we were not able to obtain such control for the networks, leading to one source of error for our validation experiments.

To prepare MPI applications for execution on the MicroGrid, the application does not need to be modified; the application is simply linked with MicroGrid libraries. A small configuration file must also be written to assist the MicroGrid in finding all necessary files during the simulation launch process. Note that the application and application configuration files can refer directly to real-world machine names or IP addresses; the MicroGrid virtualization process translates these names into appropriate physical addresses on the compute platform.

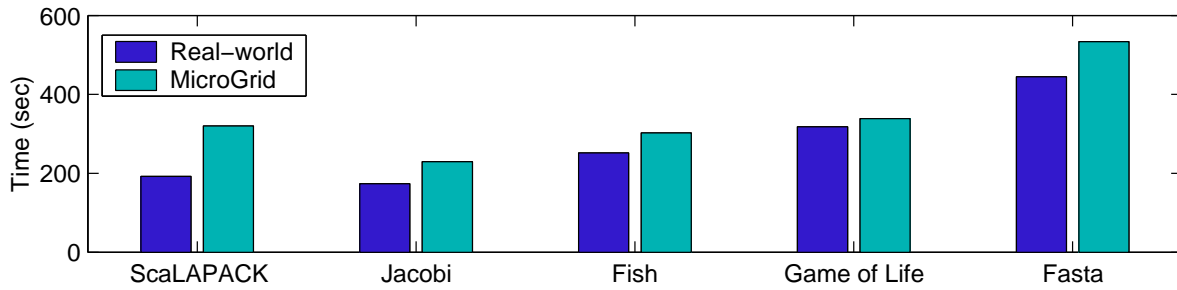


Figure 3. Comparison of application execution times on the GrADS Cluster and as measured in a MicroGrid online simulation of the GrADS Cluster.

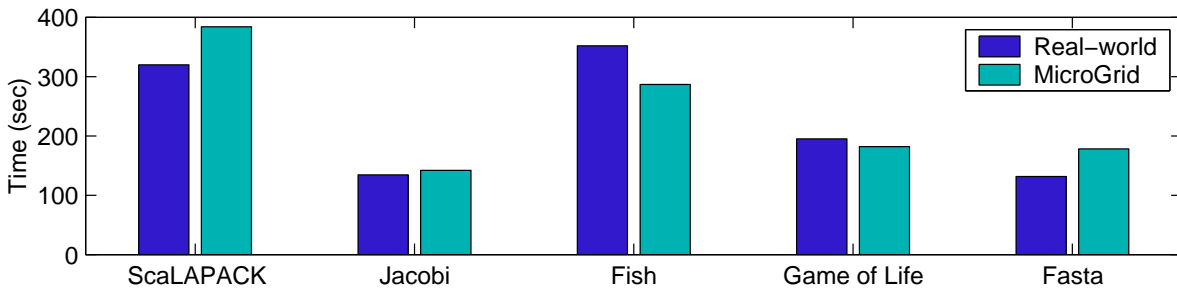


Figure 4. Comparison of application execution times on the GrADS Grid and as measured in a MicroGrid online simulation of the GrADS Grid.

	GrADS Cluster	GrADS Grid
ScaLAPACK	40.0%	16.7%
Jacobi	21.7%	5.5%
Fish	16.7%	-22.6%
Game of Life	6.2%	-9.5%
FASTA	16.7%	26.0%
Average	20.2%	16.1%

Table 1. Summary of MicroGrid prediction errors.

4.1.3. Results Validation experiments results are shown in Figure 3 for the GrADS cluster and in Figure 4 for the GrADS Grid.

Table 1 summarizes the percentage difference between the real-world execution time and that predicted by the MicroGrid online simulation. Overall, the MicroGrid provides a reasonable estimate of predicted performance, especially given the fluctuating loads and other uncertainties of execution in the real-world that can not be captured in simu-

lation. It is encouraging that the MicroGrid properly ranks the performance of all but one of the applications (the exception is ScaLAPACK) in both the grid and cluster environments. In other words, those results could inform a user as to which applications will run fastest in these environments.

For the cluster, all applications run slower on the MicroGrid than on the real testbed; most of them have error in 6%-22%, while ScaLAPACK has an error of 40%. The extra overhead comes from two major sources: 1) MaSSF has some overhead which increases network latency. 2) WrapSocket wraps many system functions for simulation, which will cause some overhead. Since ScaLAPACK involves many small messages and complicated communicated patterns, the increased network latency impact its simulation time most.

For the grid environment, the simulated time shows between 5% - 26% error. We can see several interesting differences from the cluster results. ScaLAPACK still runs slower on the MicroGrid than on the real testbed, but the times are much closer than on the cluster; this is because ScaLAPACK uses a lot of small communications and the simula-

tion overhead will have more impact on simulated LAN latency than on simulated WAN latency (as will be shown in Section 4.2). The Fish and Game Of Life applications run faster on the MicroGrid than on the real grid. In fact, in the absence of MicroGrid overheads we would expect all of the applications to run faster in the MicroGrid as we did not introduce competing load. Thus, perhaps the overhead introduced by the MicroGrid for these two applications is less significant than the presence of competing network load in the real-world, especially if the applications use large-size communications.

In general, although we plan to improve the MicroGrid validation, we feel that these initial results are promising. As users do not currently have any advance information of the application performance they can expect on new topologies, information at this level of accuracy is already useful.

4.2. High-performance Grid Architectures

In this set of experiments, we demonstrate the application of the MicroGrid for testing the performance of applications on grid architectures that are not available for real-world experimentation. This application of the MicroGrid could prove useful for a variety of reasons: the grid could exist but be unavailable to the user currently, it could be impossible to perform repeatable, controlled experiments, or the topology could even be an imagined grid architecture including higher performance components than are currently available.

In these experiments, we simulate faster and richer resources than the GrADS Grid and Cluster used in the previous section. We investigate a wide range of types and performance for networks, CPUs and storage. The immediate goal of these experiments is to discover whether these grid architectures are well suited to the five applications described in Section 3. Note that since we do not have access to real-world versions of these topologies we cannot directly validate these performance predictions. We refer to the experiments in the previous section as initial evidence that the MicroGrid provides reasonable performance predictions for these five applications.

4.2.1. Testbeds The **TeraGrid** is a large-scale grid project that combines large supercomputer style resources from across the United States of America [28]. The initial TeraGrid design, planned for operation in 2003, includes five large-scale Linux clusters at ANL, Caltech, NCSA, PSC, and SDSC. At the time of our experiments, the TeraGrid was not yet available as a production grid. We therefore simulate the TeraGrid to provide advance results.

In our experiments, we use two machines from each of the five clusters, which are connected by long wide area links (5-30ms) and multiple switch/routers and 10Gbps

	TeraGrid	Opti1	Opti2	Opti3
FASTA	8	12	4	8
Jacobi	8	9	9	9
Game of Life	8	9	9	12
Fish	10	12	12	12
ScaLAPACK	10	12	12	12

Table 2. Scheduling results on high-performance grids.

links. For the local resources, we make the following performance assumptions: each node has 3GHz CPU, 1GB memory and 1Gbps network interface, the TCP window size is 1MB. These numbers do not match the current TeraGrid specification exactly as we configured our simulation based on projected hardware specifications before the actual platform was put in place.

The Optiputer project seeks to exploit high bandwidth DWDM optical networks to support data-intensive scientific research and collaboration. The communication links are dedicated lambdas with guaranteed high bandwidth and low latency. They may run TCP or new protocols under development. We approximate the future Optiputer environment using the following network configuration.

Three sites are connected through 80Gbps switched optical backbone network; each site consists of four machines with 3GHz CPU and 2GB memory, each of them connects to the optical core router in 4Gbps. Two configurations vary the latency between the sites. **Optiputer1** has 5ms and **Optiputer2** has 60ms. To simulate the dedicated optical connections, we assume that all the nodes use infinite TCP send windows.

The **Optiputer3** models a combined Optiputer and Internet-2 grid environment. Two sites are connected through 80Gbps optical backbone network with high bandwidth and moderate (10ms) latency; each site consists of four machines with 3GHz CPUs and 2GB memory. Each machine connects to the optical core router in 4Gbps. A third site is from another backbone network with 2.4Gbps bandwidth and 40ms latency to the first two sites. This site has four machines, also with 3GHz CPU and 2GB memory, but with a 1Gbps switch and 128KB TCP window.

To decide which nodes to use to run the applications, we used an early version of grid scheduler from the GrADS project. By inputting the theory network bandwidth and latency, we got the scheduling results as in Table 2 (we only show number of selected nodes here).

4.2.2. Results Figure 5 shows the execution times measured when we ran our applications on the novel architec-

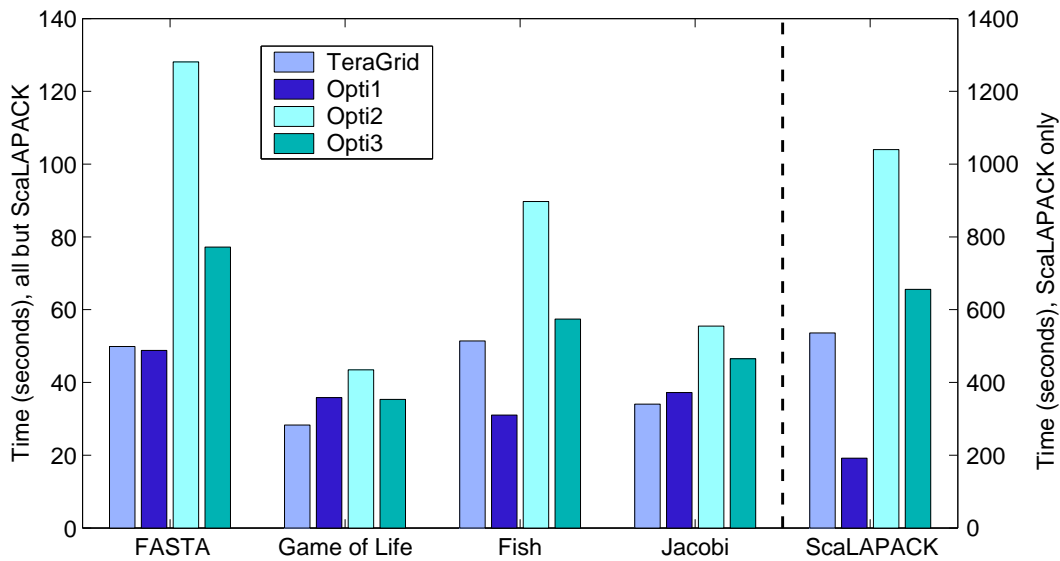


Figure 5. Forecasts of application behavior in new, high-performance grid architectures based on MicroGrid online simulations. The right-hand y-axis scale corresponds to the ScaLAPACK application only while the left-hand y-axis scale corresponds to the other four applications.

tures described above. While these quantitative results are in themselves interesting to grid users, we can make the following three interesting observations:

- Different applications run well on different architectures and the MicroGrid makes it possible for users to predict which architecture will be best for their applications. There is no single architecture that is best for all applications.
- The difference between the Optiputer1 and Optiputer2 results make it possible to directly evaluate the impact of network latencies on the application executions (in this case an increase in latency by a factor 12). For instance, this shows that while ScaLAPACK is highly impacted by network latencies, Game of Life is not.
- For the ScaLAPACK application, the communication time dominates the overall running time. This is because the application has a large number of small communications, which makes network latency the main performance factor. Furthermore, we use a relatively small problem size (6000×6000), so that the fast CPUs in the high performance testbed should not provide much benefit for the application. This is confirmed in the simulation results for ScaLAPACK, shown both in Figure 4 (for the GrADS testbed) and in Figure 5 for the four novel architectures. Indeed, the simulation results show that the application execution time is roughly proportional to the network latencies, with the

optiputer2 configuration being the worst with 60 ms latencies.

- While network latency also impacts the Fish application a lot, it is not that much significantly as on ScaLAPACK. Especially, the comparison between Figure 4 and Figure 5 shows that the application runs much faster on high-performance grids than on GrADS testbeds. This shows that Fish used both large amount of computations and large amount of communications.
- Overall, the experiments show the capability of the MicroGrid to simulate different grid environments and to run different applications on different scheduling results.

5. Related Work

Three methods have been used to perform grid experiments: real testbeds, simulation, and emulation.

Real testbeds use some specific set of real-world resources for experiments, such as PlanetLab [24], TeraGrid [28] and GrADS testbed [13]. Real testbeds provide the most realistic grid environment for experimentation. However, testbeds have many limitations: (i) fixed or limited configurations or scenarios, (ii) unpredictable and uncontrollable environments, and (iii) lack of repeatability.

Simulation is another major approach. While an important approach, the major limitations of simulation involve

the modeling step required to develop tractable simulation models. Because so little is understood about the dynamic behavior of applications much less coupled resources and networks, simulations often do not expose new phenomena, merely confirming what the designer of the experiment expected. Another problem is that network and grid simulation work has long been decoupled. Distributed applications and networks have been studied largely separately - each community employing relative simple models for the other domain. For example, several lower-level simulation models provide accurate network models [7, 2], but do not enable the network simulations to be coupled directly to applications. On the other hand, a number of software tools provide general-purpose discrete-event or even more focused Grid simulation libraries [25, 30, 21, 11, 14, 15, 8, 18]. However, these tools typically have simple models of networks and protocols - known to be inaccurate, and provide no way to couple with good network simulators.

Another alternative is network emulation [17, 32, 31]. These systems support execution of real applications over an emulated network, there are important differences between these efforts and the MicroGrid online simulator.

First, the main limitation of these emulation systems is that they do not provide global virtual time. So, they cannot control CPU speeds, and in general cannot model the arbitrary relative compute and network speeds that are possible in the MicroGrid. Such control is essential for extrapolative studies which look at technology futures and for studying extremely large systems.

Second, the network modeling in these systems is approximate. These approximations allow them to run faster (compared to MicroGrid's global synchronized simulation) at cost in fidelity and information visibility. For example, Emulab uses a set of real routers, switches and configurable software routers to emulate wide area network. ModelNet summarizes network structure as a network of queues connected by simple routing and maps the resulting network of queues onto a set of emulation cores. This summarized network is an approximation to actual detailed network behavior. The disadvantage of these approaches is that they do not allow detailed control of speed and modeling to the experiment designer. In contrast, MaSSF uses full-scale detailed packet simulation based on a distributed discrete-event simulation engine. While there have been many efforts which use PDES for network simulation, we know of no other modeling efforts that achieve detailed online network simulation of the documented scale.

6. Conclusion

The increasing acceptance of grid computing in both scientific and commercial communities presents increasing demands for an improved understanding of the performance of

applications and resources. The associations between applications and resources are no longer static, and dynamic resource sharing and application adaption introduce new complexity to the situation.

We have described a new tool, the MicroGrid, and both its validation and a demonstration of the new methodology it enables for studying performance questions involving grid applications, middleware, networks, and compute resources. These capabilities represent a wide range of new opportunities for understanding grids.

Acknowledgements

The authors would like to thank the rest of the MicroGrid and GrADS teams for their help and comments on this work. The test applications used in the paper and the experimental testbed were developed in the GrADS collaboration.

References

- [1] S. Agrawal, J. Dongarra, K. Seymour, and S. Vadhiyar. *Grid Computing: Making The Global Infrastructure a Reality*, chapter NetSolve: Past, Present, and Future - A Look at a Grid Enabled Server. John Wiley, 2003. Hey, A. and Berman, F. and Fox, G., editors.
- [2] L. Bajaj, M. Takai, R. Ahuja, K. Tang, R. Bagrodia, and M. Gerla. GloMoSim: A Scalable Network Simulation Environment. Technical Report 990027, UCLA Computer Science Department, 1999.
- [3] R. Barrett, M. W. Berry, T. F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst. *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods, 2nd Edition*. SIAM, Philadelphia, PA, 1994.
- [4] F. Berman, A. Chien, K. Cooper, J. Dongarra, I. Foster, D. Gannon, L. Johnsson, K. Kennedy, C. Kesselman, J. Mellor-Crummey, D. Reed, L. Torczon, and R. Wolski. The GrADS Project: Software support for high-level Grid application development. *International Journal of Super-computer Applications*, 15(4):327-344, 2001.
- [5] F. Berman, A. Chien, K. Cooper, J. Dongarra, I. Foster, D. Gannon, L. Johnsson, K. Kennedy, C. Kesselman, J. Mellor-Crummey, D. Reed, L. Torczon, and R. Wolski. The GrADS Project: Software support for high-level Grid application development. *International Journal of High Performance Computing Applications*, 15(4):327-344, 2001.
- [6] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. *ScalAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1997.
- [7] L. Breslau, D. Estrin, K. Fall, S. Floyd, J. Heidemann, A. Helmy, P. Huang, S. McCanne, K. Varadhan, Y. Xu, and H. Yu. Advances in network simulation. *IEEE Computer*, 33(5):59-67, 2000.

- [8] R. Buyya and M. Murshed. GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing. *The Journal of Concurrency and Computation: Practice and Experience (CCPE)*, 14(13-15), 2002.
- [9] J. Cowie, H. Liu, J. Liu, D. Nicol, and A. Ogielski. Towards realistic million-node internet simulations. In *Proceedings of the 1999 International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'99)*, June 1999. Las Vegas, Nevada.
- [10] H. Dail, F. Berman, and H. Casanova. A decoupled scheduling approach for grid application development environments. *Journal of Parallel and Distributed Computing*, 63(5):505–524, May 2003.
- [11] F. G. et al. Simkit: A high performance logical process simulation class library in c++. In *Proceedings of the 1995 Winter Simulation Conference*, 1995.
- [12] G. W. Flake. *The Computational Beauty of Nature: Computer Explorations of Fractals, Chaos, Complex Systems, and Adaptation*. MIT Press, Cambridge, MA, 1998.
- [13] The GrADS project. <http://hipersoft.cs.rice.edu/grads>.
- [14] F. Howell and R. McNab. SimJava: A Discrete Event Simulation Package for Java with Applications in Computer Systems Modelling. In *Proceedings of the First International Conference on Web-based Modelling and Simulation*, 1998.
- [15] F. Howell and M. R. SimJava: A Discrete Event Simulation Package for Java with Applications in Computer Systems Modelling. In *Proceedings of the First International Conference on Web-based Modelling and Simulation*, Jan 1998.
- [16] C. Kesselman and I. Foster. The Globus Toolkit. In C. Kesselman and I. Foster, editors, *The Grid: Blueprint for a New Computing Infrastructure*, edited by Carl Kesselman and Ian Foster, pages 259–278. Morgan Kaufmann Publishers, Inc., 1999.
- [17] T. Kielmann, H. Bal, J. Maassen, R. van Nieuwpoort, L. Eyraud, R. Hofman, and K. Verstoep. Programming environments for high-performance grid computing: the albatross project. *Future Generation Computer Systems*, 18(8), 2002.
- [18] L. Legrand, A. Marchal and H. Casanova. Scheduling Distributed Applications: The SimGrid Simulation Framework. In *Proceedings of the third IEEE International Symposium on Cluster Computing and the Grid (CCGrid'03)*, 2003.
- [19] J. Liu and D. M. Nicol. DaSSF 3.0 User's Manual, January 2001. available from <http://www.cs.dartmouth.edu/research/DaSSF/docs.html>.
- [20] X. Liu and A. Chien. Traffic-based load balance for scalable network emulation. In *Proceedings of Supercomputing Conference*, Phoenix, Arizona, November 2003.
- [21] A. Miller, R. Nair, and Z. Zhang. JSIM: A java-based simulation and animation environment. In *Proceedings of the 30th Annual Simulation Symposium (ANSS'97)*, 1997.
- [22] W. Pearson and D. Lipman. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences of the United States of America*, 85:2444–2448, 1988.
- [23] A. Petitet, S. Blackford, J. Dongarra, B. Ellis, G. Fagg, K. Roche, and S. Vadhiyar. Numerical libraries and the Grid: The GrADS experiment with ScaLAPACK. *International Journal of High Performance Computing Applications*, 15(4):359–374, 2001.
- [24] The PlanetLab Website. <http://www.planet-lab.org/>.
- [25] H. Schwetman. Csim: A c-based, process oriented simulation language. In *Proceedings of the 1986 Winter Simulation Conference*, 1986.
- [26] O. Sievert and H. Casanova. A simple MPI process swapping architecture for iterative applications. *International Journal of High Performance Computing Applications (IJHPCA)*, 2004. To appear.
- [27] H. Song, X. Liu, D. Jakobsen, R. Bhagwan, X. Zhang, K. Taura, and A. Chien. The MicroGrid: a scientific tool for modeling computational grids. In *Proceedings of Supercomputing Conference*, November 2000.
- [28] TeraGrid. <http://www.teragrid.org/>.
- [29] D. Thain, T. Tannenbaum, and M. Livny. Condor and the Grid. In *Grid Computing: Making The Global Infrastructure a Reality*. John Wiley, 2003. Hey, A. and Berman, F. and Fox, G., editors.
- [30] S. Toh. Simc: A c function library for discrete simulation. In *Proceedings of the 11th Workshop in Parallel and Distributed Simulation*, 1993.
- [31] A. Vahdat, K. Yocum, K. Walsh, P. Mahadevan, D. Kostic, J. Chase, and D. Becker. Scalability and Accuracy in a Large-Scale Network Emulator. *ACM SIGOPS Operating Systems Review*, 36, 2002.
- [32] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An Integrated Experimental Environment for Distributed Systems and Networks. In *Proc. of the 5th Symposium on Operating Systems Design and Implementation, Boston, MA, 2002*.