

Benchmark Probes for Grid Assessment

Greg Chun^{*}, Holly Dail[†], Henri Casanova^{*†} and Allan Snaveley^{*†}

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

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

gchun@cs.ucsd.edu, [hdail, casanova, allans]@sdsc.edu

Abstract

Like all computing platforms, grids are in need of a suite of benchmarks by which they can be evaluated, compared and characterized. As a first step towards this goal, we have developed a set of probes that exercise basic grid operations with the goal of measuring the performance and the performance variability of basic grid operations, as well as the failure rates of these operations. We present measurement data obtained by running our probes on a grid testbed that spans 5 clusters in 3 institutions. These measurements quantify compute times, network transfer times, and Globus middleware overhead. Our results help provide insight into the stability, robustness, and performance of our testbed, and lead us to make some recommendations for future grid development.

1. Introduction

A fundamental principle of engineering is that methods for measurement and evaluation must be incorporated into the design and development process of a system, since only that which is measured can be improved. Benchmarks have been the measurement and evaluation method of choice for computing platforms. They not only provide a means of measuring and comparing performance, but they can also serve as diagnostic tools, often highlighting the strengths and weaknesses of the platform in question. Furthermore, application developers often use benchmarks to evaluate the relative merit of application implementation paradigms. Benchmarks can therefore help guide both platform and software development.

Benchmarks have been widely used to evaluate computer architectures. To quantify the impact of technological

advances, both academic and industrial communities employ benchmark results such as those produced by the Standard Performance Evaluation Corporation (SPEC) [21]. In the High Performance Computing (HPC) realm, benchmarks can be grouped in two main categories: (i) low-level “probes” for determining the rates at which a machine can perform fundamental operations, with examples including MAPS [20], STREAM [22], PALLAS MPI benchmarks [16], and LINPACK [5]; and (ii) benchmarks meant to be representative of a class of applications, with the most popular example in HPC being the NAS Parallel Benchmarks (NPB) [4], which is based on fluid dynamics applications. Other examples include SPEC, ParkBench [12] and SPLASH [25].

Computational grids [7] have gained popularity as computing platforms that can support next generation scientific applications at unprecedented levels of scale and performance. Yet these platforms remain poorly understood with limited conventional wisdom as to their performance, stability, and usage. In particular, a key difference between grids and traditional HPC platforms is the fact that resources are inherently heterogeneous, often with dynamic levels of availability that may cause application performance fluctuations. Another key difference is that to access these platforms, applications often must utilize a middleware service layer, where some services are themselves complex distributed systems that may lead to significant overhead for some applications. In this paper, we present a set of probes for grid assessment. These probes fall into benchmark category (i) above and are designed to measure basic grid operations such as file transfers, remote execution, and queries to Grid Information Services. In this paper we provide a paper and pencil specification of the probes, we describe a reference implementation, and we present results we obtained by running the probes on a specific grid platform over the course of several weeks.

⁰This work was supported in part by NSF STI award # 0230925.

The broader scope of our work also entails the development of benchmarks in category (ii). We are currently involved in developing data-intensive benchmarks based on applications in domains such as bioinformatics [2, 18] or physics [11]. Other researchers have developed compute intensive grid benchmarks. The NAS Grid Benchmark (NGB) [8] is a grid-enabled version of the NPB and was the first available benchmark designed for the evaluation of grid platform performance. The authors actively collaborate with the NGB developers as part of the Grid Benchmarking Research Group of the Global Grid Forum (GGF) [9].

This paper is organized as follows: Section 2 describes the probes, Section 3 describes our grid testbed and experimental methodology, Section 4 presents our results, and Section 5 provides a discussion of our results and a description of future work.

2. Probe Purpose and Design

We have developed three probes for grid evaluation. With respect to the probes' design, it should first be noted that these probes are *intrusive*, and are not designed to be run year-round on the grid as is the case with lightweight resource monitoring infrastructures such as the Network Weather Service [24]. Rather, they can be used on occasion for grid verification or for predetermined periods of time to gather measurements that can be used for grid characterization research and for instantiating grid models. In this paper we emphasize the following uses of our probes: verification of basic grid functionality; measurement of performance of basic grid operations, including use of middleware; measurement of performance fluctuation for these operations; identification of performance problem areas and their sources; quantification and identification of failures and abnormalities; and generation of measurement datasets for use by grid computing researchers.

The probes are designed to run independent of a grid scheduler. Just as typical HPC benchmark probes are used to stress the most basic of operations, our probes only examine what kind of performance grid platforms are capable of achieving at a very low level. Our probes do not measure (for example) whether it is the scheduler or the underlying platform that is at fault for sluggish performance figures. Lastly, for the sake of simplicity, all grid operations are driven from a single launching host. This host alone collects timing statistics for each step of probe execution.

We provide a reference implementation of our probes, which consists of a set of Perl scripts, and uses a configuration file to contain all modifiable information such as hosts, port numbers, working directories, and other probe parameters. There are also some short C programs that perform serial computations on data files.

All probes execute the same set of initial operations:

- check for a valid grid proxy,
- perform a basic authentication to all resources involved in the probe execution,
- validate the configuration file,
- check directory sizes to ensure that target directories can accommodate files to be transferred, and
- query the MDS [3] to find available information on all nodes involved in the probe.

Included in the probe package is a C program that generates a 100 MB file that consists of repeated instances of the letters A,G,C,T,N, and X. While this data format was inspired by a typical FASTA [17] file, the composition of the file is completely arbitrary. The probes are flexible so that they can use any datafile for their execution, but for the purpose of establishing a reference configuration to standardize results in this paper, the 100 MB datafile is used exclusively in our experiments. To execute file transfers, we use GridFTP [10] via the Globus [6] program *globus-url-copy*; for launching executables we use *globus-job-run*.

The C programs use the *time.h* library and *gettimeofday()* in order to provide millisecond-level timings. The Perl scripts use the Time::HiRes Perl module when available (this package is not part of the standard Perl installation). In the absence of Time::HiRes, one can still obtain results, but only at the level of seconds.

Again, note that we have simply built *one* implementation of the probe suite that happens to use Globus, Perl, and C as underlying technologies. It is important to keep in mind that the *specification* of the probes could very easily be implemented using different languages and tools (see [19] for full detailed specifications). Having said that, we now describe each probe in detail.

2.1. 3-Node Probe

The 3-node probe transfers the 100 MB datafile from the *Database Node* to the *Compute Node*, executes a short C program on the *Compute Node* with the datafile as input, generates a results file, and then transfers that file to the *Results Node*. All file transfers for every probe in the suite are meant to be executed without employing any type of compression or other means of reducing the actual amount of data that is to be transferred over the network. The C program simply performs a number of floating-point divides for each character of the input file. There are two adjustable parameters included as part of the probe's configuration file: *compute_scale* adjusts how many floating-point divides are performed for each character of the input file, and *output_scale* adjusts how large the results file is in relation to the input file. For example, if *output_scale* is set to 3, the file output from the computation will be 300 MB. The reference probe configuration uses *compute_scale*

Cluster Name	Nodes (CPUs)	CPU Speed	CPU Type	Memory	Operating System	Network
Torc UTK	8 (2)	550 MHz	Pentium III	512 MB	Linux Red Hat 7.2	100 Mbps Switched Ethernet
MSC UTK	8 (2)	933 MHz	Pentium III	512 MB	Linux Red Hat 7.1	
Opus UIUC	16 (1)	450 MHz	Pentium II	256 MB	Linux Red Hat 7.2	100 Mbps Ethernet & 1 Gbps Switched Myrinet
Major UIUC	8 (1)	266 MHz	Pentium II	128 MB	Linux Red Hat 7.2	100 Mbps Shared Ethernet
Circus UCSD	7 (1)	1733 MHz	Athlon XP 2100+	512 MB	Linux Debian 3.0	100 Mbps Switched Ethernet

Table 1. Summary of GrADS testbed resource characteristics.

= 1 and *output_scale* = 1. A meaningful execution of this probe is one in which each node is in a different institution and/or corresponds to a different resource site. This probe is meant to mimic a data-flow application that obtains data at one site, computes a result on that data at another, and analyzes the result on a third.

2.2. Circle Probe

The circle probe takes our same 100 MB file and passes it in a ring around a given set of nodes, performing a checksum at each step along the way to ensure that the file has come across intact. As a final step, the file is transferred back to the originator, and a simple “diff” is applied to validate that the file is identical to the original. There can be any number of nodes involved in the probe. The grid testbed considered in this paper involves 5 clusters distributed over 3 institutions, and we present results for a 5-node configuration where each node is within a cluster. This probe is meant to mimic an application performing a token-passing operation around grid sites.

2.3. Gather Probe

The gather probe works in very much the same way as the 3-node probe, except that instead of transferring one datafile from a single data source, multiple datafiles are transferred in parallel to a single central node on which computation is performed. Each source datafile is 100 MB in size. As in the circle probe, there can be any number of data nodes involved, as long as they are properly noted in the configuration file. Therefore, the nodes involved in the gather probe are the *Compute Node*, *Results Node*, and a number of *Source Nodes*, numbered starting at 0. The configuration we use on our grid testbed has 3 source nodes for a total of 5 nodes (one node within each of the 5 clusters). The size of the results file that is generated from the

computation is indicated in megabytes by the *output_scale* parameter. Therefore, if *output_scale* is set to 5, the results file will be 5 MB in size. This was done in order to give the user more granular control over the results file size than in the 3-node probe. The reference configuration uses a *compute_scale* of 1, and an *output_scale* of 100 to maintain our 100 MB standard for all file transfers. This probe is meant to mimic an application that performs a file gathering operation across grid sites.

3. Experimental methodology

In this section we describe the methodology we used to conduct experiments using our probes. We developed these experiments with two goals in mind: we seek to (1) demonstrate the usability and robustness of the probes by testing them in an automated fashion for many repetitions and (2) investigate the extent to which our probes can generate datasets that provide insights into grid stability, robustness, and performance. To provide initial answers to these questions, we selected a grid testbed, identified specific probe test cases for all three probes, and ran the test cases on the chosen testbed regularly over the course of several weeks.

We chose the **grid testbed** developed for the Grid Application Development Software Project (GrADS) [1]. Table 1 provides an overview of the resources. This testbed is a collection of roughly 100 multi-purpose machines from a variety of GrADS sites: The University of California at San Diego (UCSD), The University of Tennessee at Knoxville (UTK) and The University of Illinois at Urbana-Champaign (UIUC).

3.1. Probe configurations

To obtain a large series of results that are comparable, we select a single configuration for each probe. Concerning file

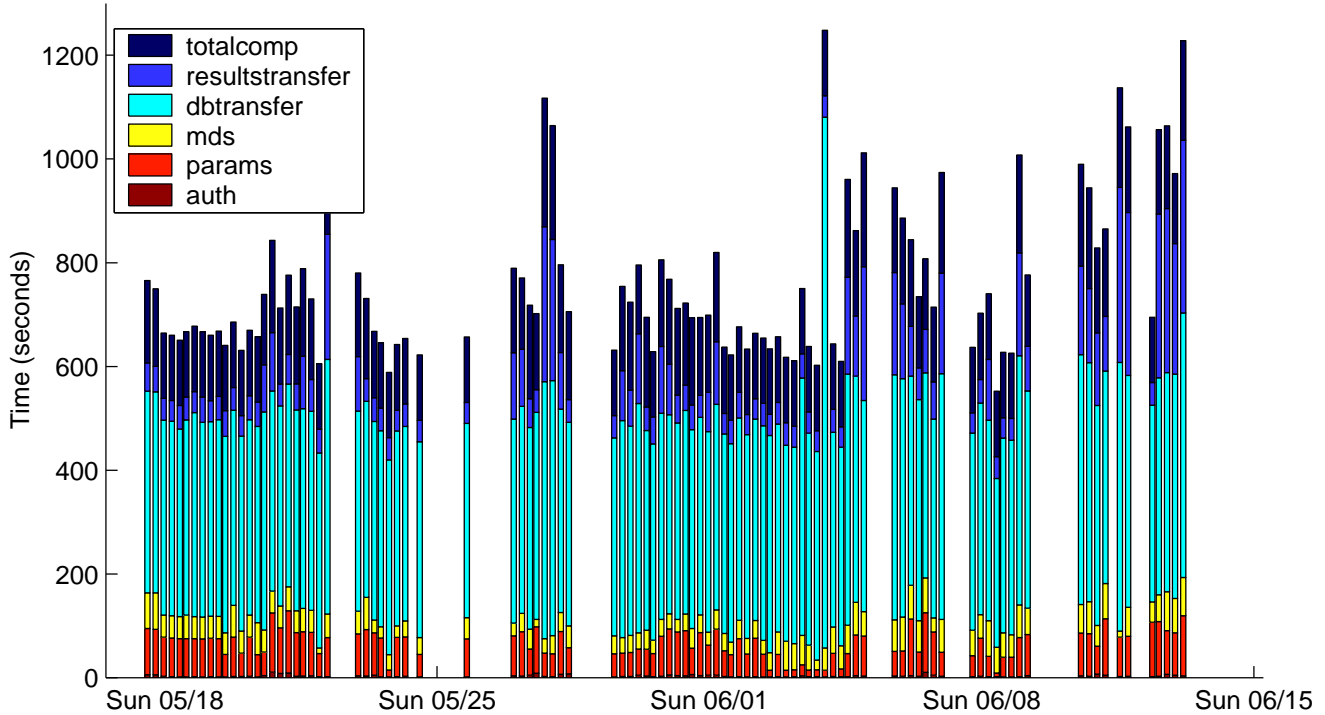


Figure 1. Measured execution times for the 3-node probe.

sizes and numbers of machines involved, we follow the reference configurations described for each probe in Section 2. Additionally, we chose the following assignments of probe roles to physical machines. These machines were chosen as representative of those available in the testbed.

For the **3-node Probe**, we mapped the *Database Node* to a Circus Cluster machine at UCSD, the *Compute Node* to an Opus Cluster machine at UIUC, and the *Results Node* to a Torc Cluster machine at UTK.

For the **Circle Probe**, we mapped *Node 0* to a Circus Cluster machine at UCSD, *Node 1* to a Torc Cluster machine at UTK, *Node 2* to a MSC Cluster machine at UTK, *Node 3* to an Opus Cluster machine at UIUC, and *Node 4* to a Major Cluster machine at UIUC. This machine topology forms a rough geographical circle around the GrADS testbed.

For the **Gather Probe** we mapped *Source 0* to a Circus Cluster machine at UCSD, *Source 1* to a Torc Cluster machine at UTK, *Source 2* to an Opus Cluster machine at UIUC, the *Compute Node* to a Circus Cluster machine at UCSD, and the *Results Node* to a Major Cluster machine at UIUC.

3.2. Experiment collection

We set up an automated system to run the probes 5 times per day and store their results in a MySQL database. Each time, the probes were run serially in the following order: 3-node, Circle, Gather; to reduce possible interactions between the probes we introduced a sleep phase of 5 minutes between each probe. We use the timings gathered through these experimental runs to generate graphs such as the one shown in Figure 1. Note that this system of gathering data and graphing the results is not part of the probe distribution itself, but is an additional tool we used to visualize and understand the data that we collected. Users of the probes are free to parse, store, and analyze the output in any way they see fit.

To obtain timings on the probe executions, we used the timing system built into the probes (see Section 2). We launched all probes from dralion.ucsd.edu; since the Perl HiRes timing module was available on this system, the probes were able to provide relatively accurate timings in our experiments.

4. Experimental Results

Figure 1 shows the results of our runs for the 3-node probe in bargraph form. The total height of each bar

represents the total execution time of the probe; time is broken down by probe activity: authentication (`auth`), validation of configuration file (`params`), query to the MDS (`mds`), input file transfer (`dbtransfer`), computation time (`totalcomp`), and output file transfer (`resulttransfer`). The gaps in the graph indicate failed probe runs, which we will explain in Section 4.1.

Figure 2 shows similar results from the circle probe. The top graph shows summary execution time information, while the lower two graphs break down transfer and checksum times for individual nodes. In general, we see variability over time for all operations and that some operations are more variable than others. Given the similarity of the graphs for the gather probe to those of the 3-node probe, they have been omitted in the interest of space.

4.1. Failures

We encountered probe failures for several different reasons. User error caused 2.8% of the probe runs to fail in that we forgot to create the proper grid proxy for a particular run. A misconfigured GridFTP server on one of the nodes caused 5.8% of the runs to fail. Expired Globus host keys were the cause of 0.3% of the runs failing. Lastly, there was one set of errors that caused 21.3% of the runs to fail that could have been the result of several different scenarios. We hypothesize that one or more of the remote sites involved in our grid testbed experienced some kind of file system problem since accompanying errors indicated that a particular directory was not able to be found. We also believe that either the problem was temporary or was fixed by the local administrators, since the error corrected itself with no intervention from us. The total percentage of failed probe runs on our testbed was 30.2%.

4.2. Variability

Tables 3(a), 3(b), and 3(c) provide summary statistics over all successful executions of each probe. Timings are presented for each section of the probes to allow study of the time required for, and variability of, different functions exercised by grid applications. Components are sorted by percentage of total running time required by that component which is displayed in the `PctTime` column. The other columns include minimum, maximum, median, and mean running times in seconds. We also calculate the standard deviation, as well as the coefficient of variation, which is the percentage given by the standard deviation divided by the mean.

All probes experienced a significant amount of variability overall, as the maximum runtime was generally between 2 and 3 times greater than the minimum. For the 3-node and circle probes, the greatest variability was seen in data trans-

fers. For the gather probe, the greatest source of variability was accessing MDS and authorization. However, since queries to the MDS did not occupy a large percentage of the total probe time, this variability did not introduce large variability in the total probe execution times. The majority of time is taken up by data transfers for all probes.

The gather probe did not seem as affected by data transfer variability as the other two. Logically, a parallel gather operation is potentially more tolerant of individual bandwidth fluctuations. That is to say, if there are three transfers progressing in parallel, it is essentially the longest transfer time that determines the total transfer time, and bandwidth variability for the two shorter transfers are of minimal consequence. Our results could be evidence to this effect. However, to be fair we must acknowledge that the gather probe may have experienced just as much variability as the 3-node probe if it were using the same node as the *Compute Node*. As we observe in Section 5, it was the operation of *writing* to a particular node at UIUC that was the source of major fluctuations. The gather probe avoided this operation based on the configuration we chose, and the issue therefore demands further investigation.

5. Discussion

Our probes were designed with the intention of representing basic operations that would be used by “data-intensive” applications, and the communication to computation ratio is clearly skewed to the communication side. Therefore, it is not surprising that bandwidth issues are the greatest source of variability in our probe runs. However, it is a fair assumption that many grid applications will be data-intensive, and so they could also exhibit substantial performance variability when run in this or similar grid environments. On traditional HPC systems, the reservation of a set of processors often guarantees the dedicated use of the communication pathways that exist between them. On the grid, this is unfortunately not the case. For applications requiring predictability, our results create an argument for the implementation of the option to obtain reserved, dedicated bandwidth between grid resources.

The probes also serve as a useful diagnostic tool for a given grid testbed. We were able to discover an incorrectly configured GridFTP server and expired Globus host keys. In addition, examination of Figure 2 reveals that a major portion of the probe’s unpredictability involves Node 3, an Opus machine at UIUC. If we look at the results from 3-Node, we also see that data transfers to UIUC take considerably longer and are fairly unpredictable as well. These observations indicate the possibility that the machines at this site represent a potential weak link with respect to predictability, and that for time-critical applications it may be wise to employ other available machines in lieu of those at

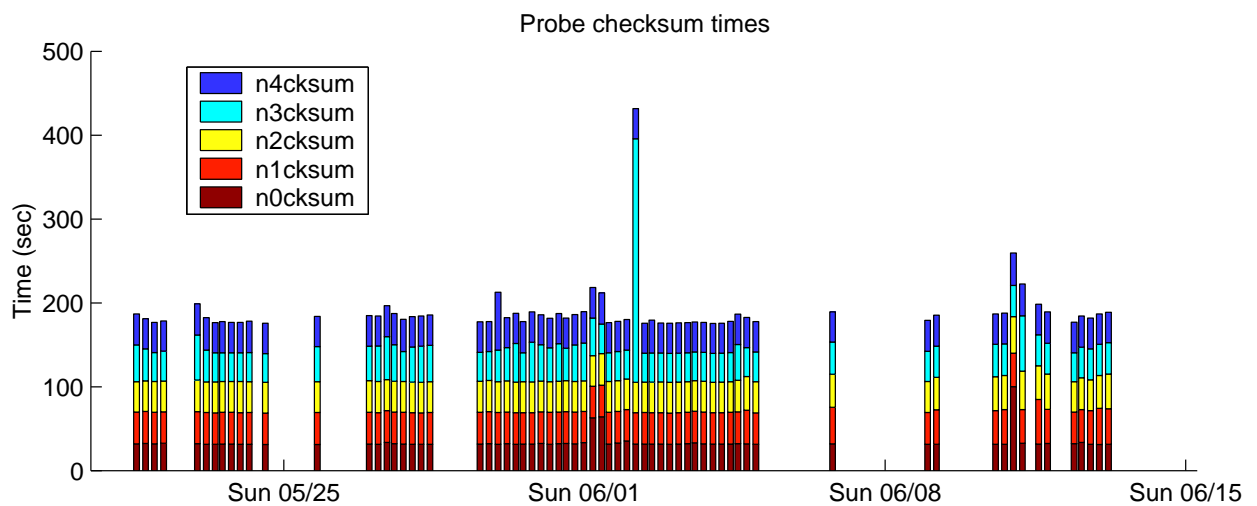
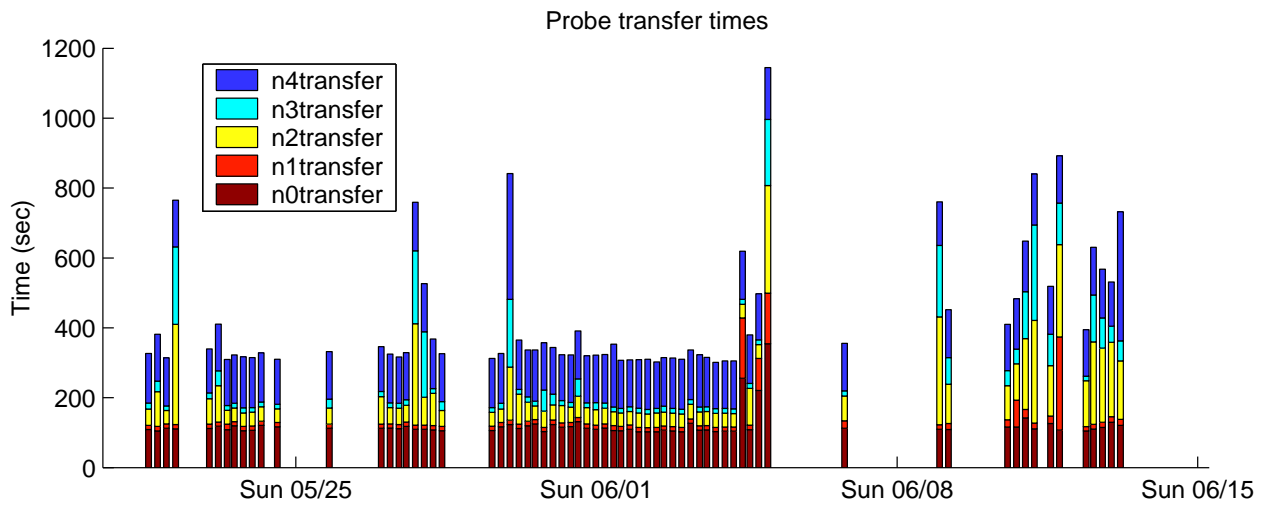
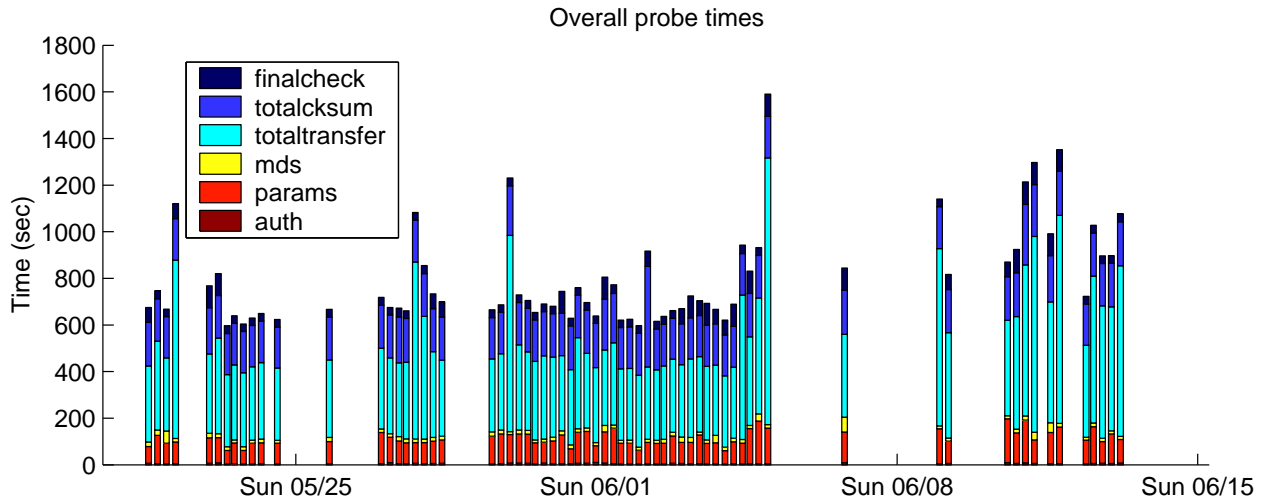


Figure 2. Measured execution times for the Circle probe.

	Min	Max	Median	Mean	Std	CoeffVar	PctTime
Data Transfer	325.4	1023.3	388.4	409.6	72.3	17.6%	53.83%
Total Comp	125.6	248.4	136.8	148.7	27.8	18.7%	19.54%
Results Transfer	38.9	337.2	49.1	92.0	79.2	86.0%	12.10%
Configuration Check	7.4	120.0	73.6	64.1	26.5	41.3%	8.43%
MDS Access	10.8	74.9	42.4	43.1	14.4	33.3%	5.67%
Authorizing	1.5	10.5	2.5	3.2	2.0	61.6%	0.42%
Total Transfer	367.6	1064.7	443.0	501.6	123.7	24.7%	65.93%
Total Time	552.3	1248.0	712.3	760.8	152.8	20.1%	100.00%

(a) Detailed 3Node Probe statistics.

	Min	Max	Median	Mean	Std	CoeffVar	PctTime
Node4 Transfer	125.1	369.4	137.4	144.7	39.3	27.1%	18.16%
Node0 Transfer	102.7	354.4	110.9	119.7	36.9	30.8%	15.02%
Configuration Check	57.7	187.7	98.4	108.5	29.8	27.5%	13.62%
Node2 Transfer	38.2	308.5	46.6	89.6	79.4	88.6%	11.24%
Final Check	32.3	100.3	33.8	51.8	25.7	49.5%	6.51%
Node3 Transfer	13.0	272.9	14.7	47.0	63.2	134.5%	5.90%
Node3 Checksum	34.2	290.2	36.9	42.3	31.0	73.3%	5.31%
Node1 Checksum	37.1	53.4	37.8	38.5	2.3	6.0%	4.84%
Node2 Checksum	36.2	45.7	36.7	37.4	1.8	4.9%	4.70%
Node4 Checksum	35.1	68.9	36.2	37.0	4.0	10.9%	4.64%
Node0 Checksum	31.2	100.0	31.7	33.8	9.8	29.0%	4.24%
Node1 Transfer	11.8	266.4	12.4	23.4	40.6	173.6%	2.93%
MDS Access	12.7	63.8	15.5	18.0	8.9	49.3%	2.26%
Authorizing	2.8	10.0	4.5	5.1	1.9	37.3%	0.64%
Total Transfer	301.1	1144.5	336.7	424.4	179.9	42.4%	53.25%
Total Checksum	175.5	431.6	182.4	189.1	32.7	17.3%	23.72%
Total Time	596.8	1590.0	712.0	797.0	206.9	26.0%	100.00%

(b) Detailed Circle Probe statistics.

	Min	Max	Median	Mean	Std	CoeffVar	PctTime
Data Transfer	162.8	323.5	196.3	211.8	39.9	18.8%	42.83%
Results Transfer	104.6	384.7	111.0	120.2	36.4	30.3%	24.32%
Configuration Check	19.6	183.1	82.6	85.1	25.0	29.4%	17.21%
Total Comp	31.1	96.4	31.6	43.6	17.4	39.9%	8.82%
MDS Access	11.7	72.3	24.0	29.8	15.4	51.5%	6.02%
Authorizing	2.2	11.9	3.3	3.9	1.9	47.2%	0.79%
Total Transfer	277.6	667.1	313.3	332.0	61.4	18.5%	67.15%
Total Time	376.8	807.9	472.1	494.5	78.1	15.8%	100.00%

(c) Detailed Gather Probe statistics.

Figure 3. Summary statistics over all successful probe runs.

UIUC. While this use of the probes was not our primary focus at the onset of the project, it proved to be valuable and we are currently collaborating with the TeraGrid [23] development team to integrate our probes with their evaluation and testing infrastructure [13].

Finally, repeated runs of our probes generate datasets that can be used as a basis for grid computing research that is based on quantitative data. This data can be analyzed at a coarse level to obtain general characterization of grid platform properties. For instance, on our testbed, we observed that the variability of probe run execution time is between 100% and 200%, and our probes make it straightforward to identify the sources of that variability. Such information can be used to evaluate quantitatively the potential benefit of techniques such as bandwidth reservation and adaptive scheduling for applications that are sensitive to variability, etc. Measurement series obtained with our probes can also be analyzed more finely to identify trends and statistical properties that can be used to instantiate abstract grid models that are representative of existing production grids. In general, the availability of long-range measurement datasets is the foundation of thorough scientific investigation and the grid computing field lacks such datasets.

While the results presented in this paper are an important first step, they are limited in scope (one testbed) and time-scale (one month). We envision that longer runs on multiple grids will generate invaluable data for advancing the scientific study of grid computing technology and platforms. In this view, we are currently deploying our probes on more grid platforms, namely the TeraGrid [23], NASA's IPG [14], and the NPACI Grid [15]. We will report on extensive measurements and conduct thorough analysis and characterization of these measurements in an upcoming paper. The probes are available for download from <http://grail.sdsc.edu/projects/grasp>.

References

- [1] 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. *IJSA*, 15(4):327–344, 2001.
- [2] The Biomedical Informatics Research Network (BIRN). <http://birn.ncrr.nih.gov>.
- [3] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid information services for distributed resource sharing. In *Proceedings of the 10th IEEE Symposium on High-Performance Distributed Computing*, August 2001.
- [4] D. H. Bailey and E. Barszcz and J. T. Barton and D. S. Browning and R. L. Carter and D. Dagum and R. A. Fatoohi and P. O. Frederickson and T. A. Lasinski and R. S. Schreiber and H. D. Simon and V. Venkatakrisnan and S. K. Weeratunga. The NAS Parallel Benchmarks. *The International Journal of Supercomputer Applications*, 5(3):63–73, 1991.
- [5] J. Dongarra, J. Bunch, C. Moler, and G. W. Stewart. LINPACK users guide. Technical report, 1979. <http://www.top500.org/lists/linpack.php>.
- [6] I. Foster and C. Kesselman. The Globus Project: A status report. In *Proceedings of the 7th Heterogeneous Computing Workshop*, pages 4–18. IEEE Press, 1998.
- [7] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, Inc., 1999.
- [8] M. Frumkin and R. Van der Wijngaart. NAS Grid Benchmarks: A Tool for Grid Space Exploration. *Cluster Computing*, 5(3), 2002.
- [9] Global Grid Forum. <http://www.gridforum.org>.
- [10] GridFTP: Universal Data Transfer for the Grid, Globus whitepaper. Available at <http://www.globus.org/datagrid/gridftp.html>.
- [11] GriPhyN - Grid Physics Network. <http://www.griphyn.org/>.
- [12] T. Hey and D. Lancaster. The development of Parkbench and performance prediction. *The International Journal of High Performance Computing Applications*, 14(3):205–215, 2000.
- [13] TeraGrid Inca Test Harness and Reporting Framework. <http://tech.teragrid.org/inca/>.
- [14] NASA's Information Power Grid (IPG). <http://www.ipg.nasa.gov>.
- [15] The NPACI Grid. <http://npacigrid.npaci.edu/>.
- [16] The PALLAS MPI Benchmarks. <http://www.pallas.com>.
- [17] 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.
- [18] J. Ponaramenko, I. Shindyalov, and P. Bourne. Building an automated classification of dna-binding protein domains. *Bioinformatics*, 18(Suppl. 2):S192–S201, 2002.
- [19] Probe Paper-and-Pencil Specifications. http://grail.sdsc.edu/projects/grasp/probes/paper_and_pencil.pdf.
- [20] A. Snaveley, L. Carrington, N. Wolter, J. Labarta, R. Badia, and A. Purkayastha. A framework for application performance modeling and prediction. In *Proceedings of Supercomputing*, November 2002.
- [21] The SPEC Benchmarks. <http://www.specbench.org>.
- [22] STREAM: Measuring Sustainable Memory Bandwidth in High Performance Computers. <http://www.cs.virginia.edu/stream/>, 1995.
- [23] The TeraGrid Project. <http://www.teragrid.org>.
- [24] R. Wolski, N. T. Spring, and J. Hayes. The Network Weather Service: A distributed resource performance forecasting service for metacomputing. *The Journal of Future Generation Computing Systems*, 15(5-6):757–768, 1999.
- [25] S. C. Woo, M. Ohara, E. Torrie, J. P. Singh, and A. Gupta. The SPLASH-2 Programs: Characterization and methodological considerations. In *Proceedings of the 22nd International Symposium on Computer Architecture*, Santa Margherita Ligure, Italy, June 1995.