Benchmark Probes for Grid Assessment

Greg Chun*, Holly Dail†, Henri Casanova*† and Allan Snavely†

*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 and characterized. As a first step towards this goal, we have developed a set of probes that exercise basic grid operations by simulating simple grid applications. We run these probes on a testbed, collecting performance data such as compute times, network transfer times, and Globus middleware overhead. The results of our experiments help provide insight into the stability, robustness, and performance of this grid testbed, as well as some recommendations for future grid development.

I. Introduction

Benchmarks allow us to quantify and understand capabilities of computing platforms. They not only provide a means of comparing performance, but they also serve as diagnostic tools, often highlighting the strengths and weaknesses of the platform in question. Application developers and system architects both use benchmarks to reason about the relative performance of systems. Benchmarks therefore help guide both platform and software development.

Benchmarks have been widely used as a means for comparing 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) [19]. In the High Performance Computing (HPC) realm, benchmarks can be classified into two main categories: (i) low-level "probes" for determining the rates at which a machine can perform fundamental operations, with examples including MAPS [18], STREAM [20], PALLAS [15] MPI benchmarks, and LINPACK [5]; and (ii) representative applications meant to reflect the computational needs 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 [13] and SPLASH [22].

Computational grids [7, 8] are gaining in popularity as scientific platforms, yet remain poorly understood with limited conventional wisdom as to their performance, stability, and usage. In this paper, we present a set of probes for grid assessment. These probes fall into benchmark category (i) and are designed to measure basic grid operations such as file transfers, remote execution, and queries to Grid Information Services. The broader scope of our work also entails the development of benchmarks in category (ii). Other researchers have developed compute intensive grid benchmarks. The NAS Grid Benchmark (NGB) [9] is a grid-enabled version of the NPB and was the first available benchmark targeted for evaluating grid platform performance. We are focusing on data-intensive benchmarks based on applications in domains such as bioinformatics [2, 17] or physics [12]. These benchmarks are still in development; the authors actively collaborate with the NGB developers as part of the Grid Benchmarking Research Group of the Global Grid Forum (GGF) [10].

II. Probe Design

We have developed three probes for grid evaluation. They have been implemented as Perl scripts, and use 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 datafiles. All probes

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

execute the same set of initial operations:

- Check for a valid grid proxy
- Perform a basic authentication to all nodes involved in the probe
- Validate the configuration file, if necessary
- Check directory sizes to ensure that target directories can accomodate files to be transferred
- 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 [16] 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, the 100 MB datafile is used exclusively in our experiments. To execute file transfers, we use GridFTP [11] 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, although 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. We now describe each probe in detail.

A. 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 called "compute" on the *Compute Node* with the datafile as input, generates a results file, and then transfers that file to the *Results Node*. The compute 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* = 1 and *output_scale* = 1. This probe is meant to mimic a pipelined application that obtains data at one site, computes a result on that data at another, and analyzes the result on a third.

B. 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, but we have chosen 5 nodes to be the reference configuration. This probe is meant to emulate an application performing a token-passing operation around grid sites.

C. 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 notated 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 0 through x. Our reference configuration has 3 sources for a total of 5 nodes. 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 greater 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 emulate an application that performs a gather operation across grid sites.

III. Experimental methodology

In this section we describe the methodology we used to conduct experiments using the grid probes. We developed these experiments with two goals in mind:

- Demonstrate the usability and robustness of the probes by testing them in an automated fashion for many repetitions.
- Investigate the extent to which our probes can provide insights into grid stability, robustness, and performance.

To provide initial answers to these questions, we selected a testbed for initial study, 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.

A. Testbed

For our initial investigations, we chose a testbed developed for the Grid Application Development Software Project (the GrADS testbed) [1]. 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), The University of Illinois at Urbana Champagne (UIUC), and the University of Houston (UH). Figure 1 and Table 1 provide an overview of the testbed; UH machines are not included in these descriptions because we did not target them for this work.

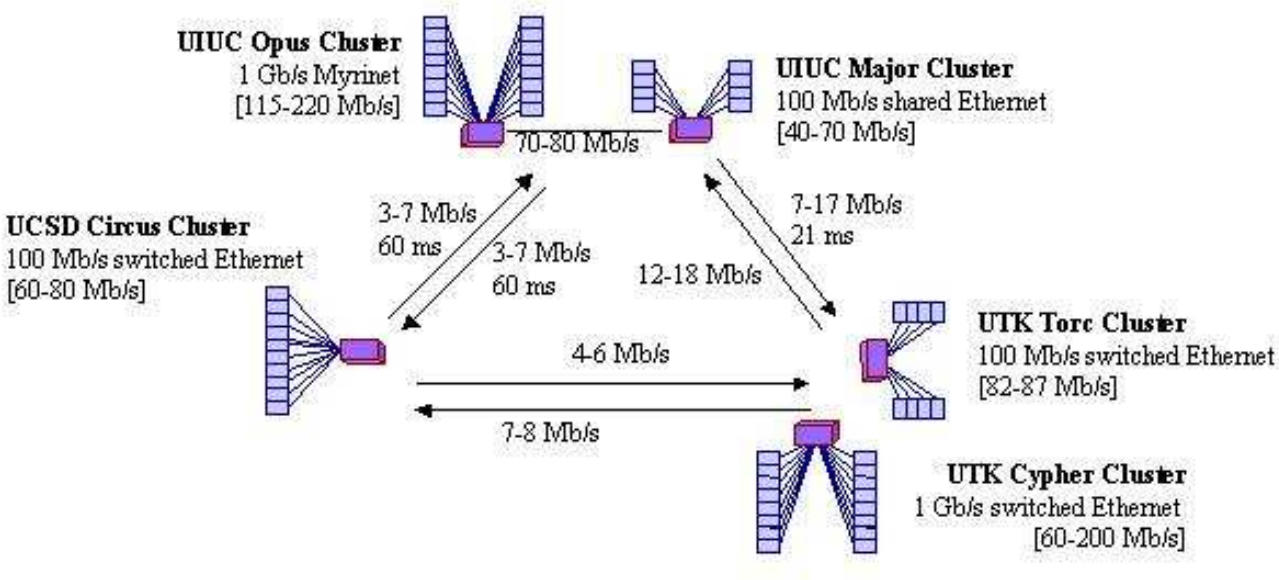

Fig. 1. A snapshot of testbed resources. Network links are labeled with available bandwidth in megabits per second; these values were collected in May, 2003 by Network Weather Service sensors.

B. Probe configurations

To obtain a large series of results that are comparable, we select a single configuration for each probe. Concerning file sizes and numbers of machines involved, we follow the reference configurations described for each probe in Section II. 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 **3node Probe**, we mapped the *Database Node* to a Circus Cluster machine at UCSD (dralion.ucsd.edu - a 1733 MHz Athlon), the *Compute Node* to an Opus Cluster machine at UIUC (opus13-m.cs.uiuc.edu) and the *Results Node* to a Torc Cluster machine at UTK (torc3.cs.utk.edu).

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) 1 (1) 1 (1) 1 (1)	1733 MHz 1500 MHz 700 MHz 700 MHz	Athlon XP 2100+ Athlon XP 1800 Athlon Athlon	512 MB 256 MB 256 MB 384 MB	Linux Debian 3.0	100 Mbps Switched Ethernet

Table 1. Summary of GrADS testbed resource characteristics.

For the **Circle Probe**, we mapped *Node 0* to a Circus Cluster machine at UCSD (quidam.ucsd.edu - a 1733 MHz Athlon), *Node 1* to a Torc Cluster machine at UTK (torc4.cs.utk.edu), *Node 2* to a MSC Cluster machine at UTK (msc01.cs.utk.edu), *Node 3* to a Opus Cluster machine at UIUC (opus5-m.cs.uiuc.edu), and *Node 4* to a Major Cluster machine at UIUC (dmajor.cs.uiuc.edu). 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 (quidam.ucsd.edu), *Source 1* to a Torc Cluster machine at UTK (torc4.cs.utk.edu), *Source 2* to a Opus Cluster machine at UIUC (opus4-m. cs.uiuc.edu), the *Compute Node* to a Circus Cluster machine at UCSD (saltimbanco.ucsd.edu - a 1733 MHz Athlon), and the *Results Node* to a Major Cluster machine at UIUC (dmajor.cs.uiuc.edu).

C. Experiment collection

We set up an automated system to run the probes 5 times per day. Each time, the probes were run serially in the following order: 3node, Circle, Gather; to reduce possible interactions between the probes we introduced a sleep phase of 5 minutes between each probe.

To obtain timings on the probe executions we used the timing system built into the probes (see Section II). 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.

IV. Experimental Results

A. Raw Results

Figure 2 show 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. The gaps in the graph indicate failed probe runs, which we will explain in Section IV-B.

Figure 3 shows the 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 between runs, and certain operations are more volatile 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.

B. 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

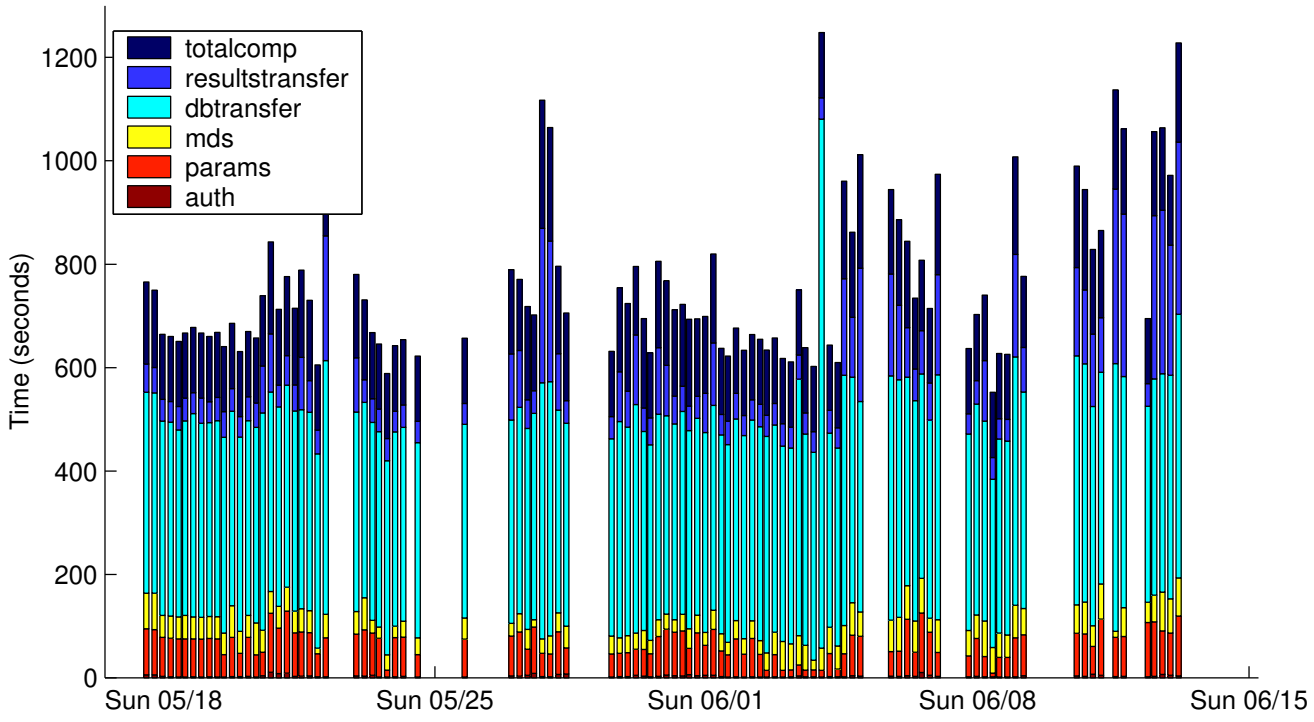

Fig. 2. Measured execution times for the 3-node probe.

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 filesystem 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 was 30.2%.

C. Variability

Tables 4(a), 4(b), and 4(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 transfers. For the gather probe, the greatest source of variability was accessing MDS and authorization. However, given what little percent of the total probe time is usually taken up by querying MDS and authorization, the overall effect is fairly negligible. 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 V, 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.

V. Discussion

Our probes were designed with the intention of representing applications of a "data-intensive" nature. The communication to computation ratio is clearly skewed to be heavier on 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 considered 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 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. We were able to discover an incorrectly configured GridFTP server and expired Globus host keys. In addition, examination of Figure 3 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 UIUC. However, the cause for this behavior could also be, as we discovered after some investigation, that we were using a publicly-accessible myrinet interface to the UIUC machines which was not intended to be used for this type of activity. In either case, we learn something valuable about our grid platform.

VI. Future Work and Conclusion

We plan to extend our current work to develop a suite of "application benchmarks" that more closely mimic the activity of real-world grid applications. Since grid applications and platforms are still in their early stages, one of the most significant challenges is to determine what types of applications need to be represented in the suite. We will use information gathered from interviews with researchers currently making use of grid technology to help guide our development. We also plan to continue collecting data from probe runs on other grid platforms. Current candidates include the TeraGrid [21] and NASA's IPG [14].

Our work thus far already shows promise as to the utility of grid probes and benchmarks. We use them as diagnostic tools for individual grid platforms, but they also bring to light possible areas of focus for grid development itself. It is our hope that our final suite will aid both grid users and developers alike as the technology of the grid continues to evolve.

References

[1] Berman, F., Chien, A., Cooper, K., Dongarra, J., Foster, I., Gannon, D., Johnsson, L., Kennedy, K., Kesselman, C., Mellor-Crummey, J., Reed, D., Torczon, L., and Wolski, R. The GrADS Project: Software support for high-level Grid application development. *IJSA 15*, 4 (2001), 327–344.

[2] The Biomedical Informatics Research Network (BIRN). `http://birn.ncrr.nih.gov`.

[3] Czajkowski, K., Fitzgerald, S., Foster, I., and Kesselman, C. 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. Venkatakrishnan and S. K. Weeratunga. The NAS Parallel Benchmarks. *The International Journal of Supercomputer Applications 5*, 3 (1991), 63–73.

[5] Dongarra, J., Bunch, J., Moler, C., and Stewart, G. W. LINPACK users guide. Tech. rep., 1979. `http://www.top500.org/lists/linpack.php`.

[6] Foster, I., and Kesselman, C. The Globus Project: A status report. IEEE Press, pp. 4–18.

[7] Foster, I., and Kesselman, C., Eds. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, Inc., 1999.

[8] Foster, I., Kesselman, C., and Tuecke, S. The Anatomy of the Grid: Enabling scalable virtual organizations. *International Journal of Supercomputer Applications 15*, 3 (2001), 200–222.

[9] Frumkin, M., and Van der Wijngaart, R. NAS Grid Benchmarks: A Tool for Grid Space Exploration. *Cluster Computing 5*, 3 (2002).

[10] Global Grid Forum. `http://www.Gridforum.org`.

[11] GridFTP: Universal Data Transfer for the Grid, Globus whitepaper. Available at http://www.globus.org/datagrid/gridftp.html.

[12] GriPhyN - Grid Physics Network. `http://www.griphyn.org/`.

[13] Hey, T., and Lancaster, D. The development of Parkbench and performance prediction. *The International Journal of High Performance Computing Applications 14*, 3 (2000), 205–215.

[14] NASA's Information Power Grid (IPG). `http://www.ipg.nasa.gov`.

[15] The PALLAS MPI Benchmarks. `http://www.pallas.com`.

[16] Pearson, W., and Lipman, D. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences of the United States of America 85* (1988), 2444–2448.

[17] Ponaramenko, J., Shindyalov, I., and Bourne, P. Building an automated classification of dna-binding protein domains. *Bioinformatics 18*, Suppl. 2 (2002), S192–S201.

[18] Snavely, A., Carrington, L., Wolter, N., Labarta, J., Badia, R., and Purkayastha, A. A framework for application performance modeling and prediction. In *Proceedings of Supercomputing* (November 2002).

[19] The SPEC Benchmarks. `http://www.specbench.org`.

[20] STREAM: Measuring Sustainable Memory Bandwidth in High Performance Computers. `http://www.cs.virginia.edu/stream/`, 1995.

[21] The TeraGrid Project. `http://www.teragrid.org`.

[22] Woo, S. C., Ohara, M., Torrie, E., Singh, J. P., and Gupta, A. The SPLASH-2 Programs: Characterization and methodological considerations. In *Proceedings of the 22nd International Symposium on Computer Architecture* (Santa Margherita Ligure, Italy, June 1995).

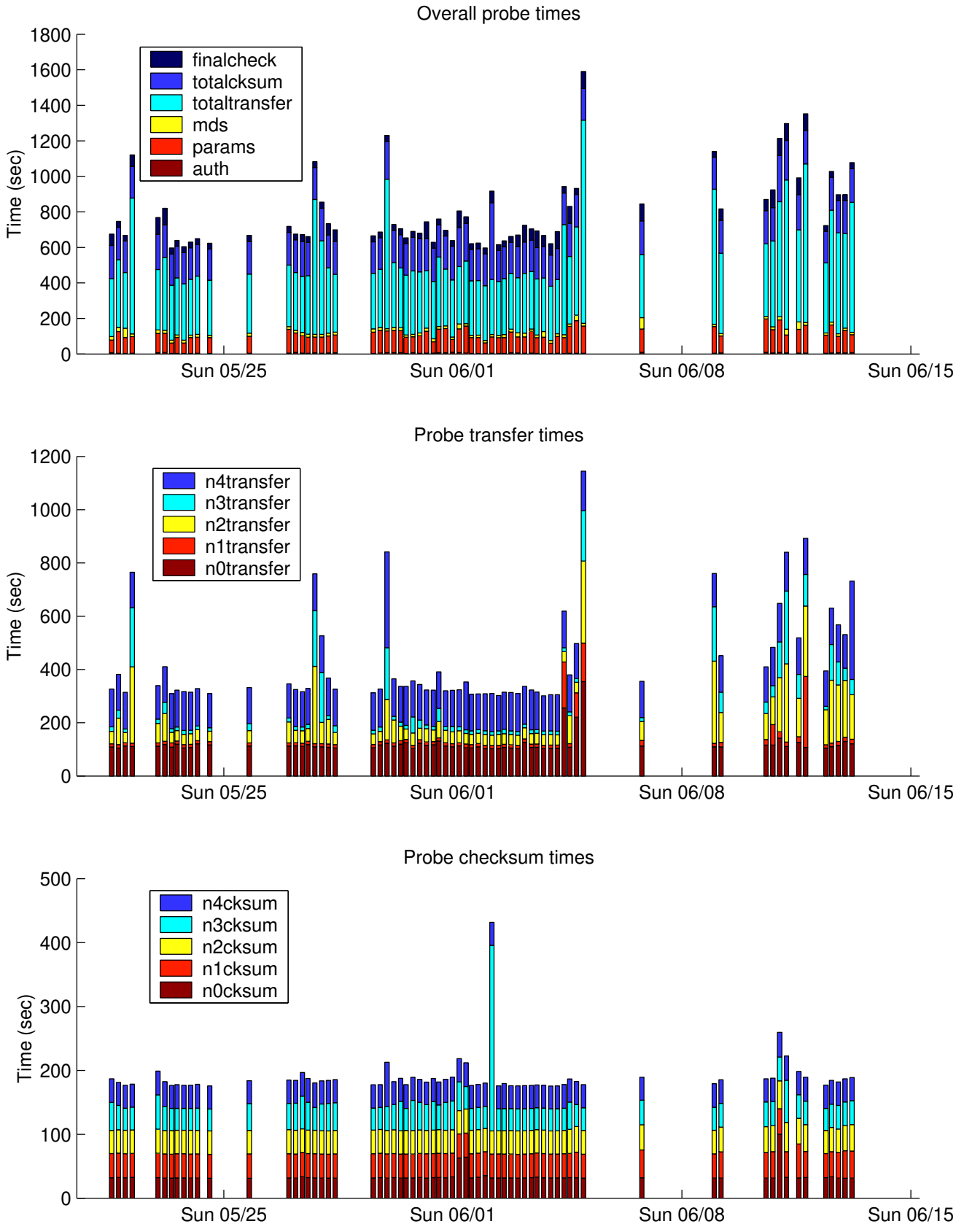

Fig. 3. Measured execution times for the Circle probe.

	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.

Fig. 4. Summary statistics over all successful probe runs.