NETLOGGER

A Toolkit for Distributed System Performance Tuning and Debugging

Dan Gunter (dkgunter@lbl.gov), Brian Tierney
(bltierney@lbl.gov)
Lawrence Berkeley National Laboratory, 1 Cyclotron Road,
Berkeley, CA 94720

Abstract: Developers and users of high-performance distributed systems often observe performance problems such as unexpectedly low throughput or high latency. Determining the source of the performance problems requires detailed end-to-end instrumentation of all components, including the applications, operating systems, hosts, and networks. In this paper we describe a methodology that enables the real-time diagnosis of performance problems in complex high-performance distributed systems. The methodology includes tools for generating timestamped event logs that can be used to provide detailed end-to-end application and system level monitoring; and tools for visualizing the log data and real-time state of the distributed system. This methodology, called NetLogger, has proven invaluable for diagnosing problems in networks and in distributed systems code. This approach is novel in that it combines network, host, and application-level monitoring, providing a complete view of the entire system. NetLogger is designed to be extremely lightweight, and includes a mechanism for reliably collecting monitoring events from multiple distributed locations.

Key words: distributed systems performance analysis and debugging

1. INTRODUCTION

The performance characteristics of distributed applications are complex, rife with “soft failures” in which the application produces correct results but has much lower throughput or higher latency than expected. Bottlenecks can occur in any component along the data's path: applications, operating systems, device drivers, network adapters, and network components such as switches and routers. We have developed a methodology, known as NetLogger (short for *Net*worked Application

Logger), for monitoring, under realistic operating conditions, the behavior of all the elements of the application-to-application communication path in order to determine exactly what is happening within a complex system.

Distributed application components, as well as some operating system components, are modified to perform precision timestamping and logging of "interesting" events, at every critical point in the distributed system. The events are time-correlated with the system's behavior in order to characterize the performance of all aspects of the system and network in detail during actual operation.

NetLogger has demonstrated its usefulness in a variety of contexts, but most frequently in loosely coupled client-server architectures. We began developing NetLogger in 1994, and in previous work we have shown that detailed application monitoring is vital for both performance analysis and application debugging [6]. This paper gives a very brief summary of the main NetLogger components and provides a case study. A longer version of this paper that includes extended descriptions, details on recent NetLogger enhancements, and a more complete set of references, is [6].

There are a number of systems that address application monitoring. *log4j*, part of the Apache Project [4], has produced a flexible library for Java application logging. However, the performance of *log4j* is far lower than is necessary for detailed monitoring.

Another instrumentation package is the Open Group's Enterprise Management Forum's [5] Application Response Measurement (ARM) API, which defines function calls that can be used to instrument an application for transaction monitoring. Tools to visualize and discover patterns of ARM events are described in [3].

2. NETLOGGER TOOLKIT COMPONENTS

The NetLogger Toolkit consists of four components: an API and library of functions to simplify the generation of application-level event logs, a set of tools for collecting and sorting log files, a set of host and network monitoring tools, and a tool for visualization and analysis of the log files. Instrumentation is performed by modifying source code and linking with the NetLogger library. All the tools in the NetLogger Toolkit share a common log format, and assume the existence of accurate and synchronized system clocks. We have found that the NTP tools that ship with most Unix systems (e.g.: *ntpd*) can often provide the desired level of synchronization.

Figure 1 shows sample results from the NetLogger Visualization tool, *nlv*, using a remote data copy application. The events being monitored are shown on the y-axis, and time is on the x-axis. From bottom to top, one can see CPU utilization events, application events, and TCP retransmit events all on the same graph. Each semi-vertical line represents the "life" of one block of data as it moves through the application. The gap in the middle of the graph, where only one set of header and data blocks are transferred in three seconds, correlates exactly with a set of TCP retransmit events. Thus, this plot makes it easy to see that the "pause" in the transfer is due to TCP retransmission errors on the network.

We have found exploratory, visual analysis of the log event data (as opposed to rule-based correlation such as that presented in [3]) to be the most useful means of determining the causes of performance anomalies. The NetLogger Visualization tool, *nlv*, has been developed to provide a flexible and interactive graphical representation of system-level and application-level events. For more details, see [7].

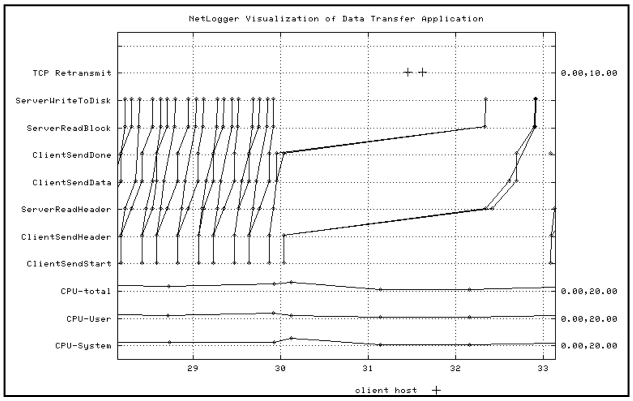

Figure 1. NetLogger Visualization Tool

NetLogger events can be formatted as an easy to read and parse ASCII format. To address the overhead issues discussed above, NetLogger includes a highly efficient self-describing binary wire format, capable of handling over 600,000 events per second. NetLogger also includes a remote activation mechanism, and reliability support.

3. CASE STUDIES

Note: due to space limitations, the figures illustrating these two case studies are online at http://www-didc.lbl.gov/NetLogger/examples/ *under* radiance_pic.png *and* globus-logs/gridftp_select_bug.png *for the first and second case study, respectively.*

In the first case study, NetLogger was used to instrument a 3-dimensional visualization engine called Radiance [2] that read data off disk, rendered it, and sent it out to clients for display. The *lifelines* in these graphs represent the data flow to generate one image. The upper graph shows the results before NetLogger tuning. The developer in this case had assumed that the I/O time was greater than the image rendering time, and therefore didn't bother to make the program multi-threaded and overlap processing with I/O. After seeing these results, however, the developer made the program multi-threaded. The new code produced the results in the lower graph; almost double the performance.

In the second case study a high-performance FTP client/server called GridFTP [1] was instrumented. Among other enhancements, GridFTP extends the FTP protocol to transfer a single file across several parallel TCP streams. In some WAN environments this can cause a dramatic (almost linear) speedup.

The bottom three groups of lifelines show headers and packets arriving on three sockets for a parallel FTP client. Data should be steadily arriving on all three sockets, but clearly the client was not servicing all three sockets equally. Further analysis showed that there was a months-old bug in the way the Unix *select*() call was being used. Despite the bug, the multi-stream version of the FTP client was faster than the single stream version, so no one had noticed this problem. This is the type of subtle bug that NetLogger is very good at tracking down.

These two case studies demonstrate the NetLogger's ability to analyze a single application. In both cases *nlv* made it easy to spot problems. However, NetLogger's real power is demonstrated by analyzing a distributed application, and time-correlating monitoring from the application, host, and network.

4. CONCLUSIONS

In order to achieve high end-to-end performance in widely distributed applications, a great deal of analysis and tuning is needed. The top-to-bottom, end-to-end approach of NetLogger has proven to be a very useful mechanism for analyzing the performance of distributed applications in high-speed wide-area networks. All NetLogger Toolkit components under an Open Source license, and can be downloaded from http://www-didc.lbl.gov/NetLogger/.

ACKNOWLEDGMENTS

This work was supported by the Director, Office of Science. Office of Advanced Scientific Computing Research. Mathematical, Information, and Computational Sciences Division under U.S. Department of Energy Contract No. DE-AC03-76SF00098. This is report no. LBNL-51276.

REFERENCES

[1] Allcock B., Bester, J., Bresnahan, J., Chervenak, A., Foster, I., et.al. *Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing*. IEEE Mass Storage Conference, 2001.

[2] Bethel, W., B. Tierney, J. Lee, D. Gunter, S. Lau. *Using High-Speed WANs and Network Data Caches to Enable Remote and Distributed Visualization*. Proceeding of the IEEE Supercomputing 2000 Conference, Nov. 2000.

[3] Burns, L., JL Hellerstein, S Ma, CS Perng, DA Rabenhorst, D Taylor, *A Systematic Approach to Discovering Correlation Rules for Event Management*, IFIP/IEEE International Symposium on Integrated Network Management, 2001.

[4] log4j: http://jakarta.apache.org/log4j/docs/index.html

[5] Open Group, Enterprise Management Forum. 2002, http://www.opengroup.org/management/arm.htm.

[6] Tierney, B., W. Johnston, B. Crowley, G. Hoo, C. Brooks, D. Gunter. *The NetLogger Methodology for High Performance Distributed Systems Performance Analysis*. Proceeding of IEEE High Performance Distributed Computing, July 1998, LBNL-42611. http://www-didc.lbl.gov/NetLogger/

[7] Tierney, B. and D. Gunter, NetLogger: A Toolkit for Distributed System Performance Tuning and Debugging , LBNL Tech Report LBNL-51276. http://www-didc.lbl.gov/papers/NetLogger.overview.pdf