

End-to-End Data Solutions for Distributed Petascale Science

Jennifer M. Schopf^{1,2}, Ann Chervenak³, Ian Foster^{1,2,4}, Dan Fraser^{1,2},
Dan Gunter⁵, Nick LeRoy⁶, Brian Tierney⁵

¹ Computation Institute, University of Chicago and Argonne National Laboratory

² Mathematics and Computer Science Division, Argonne National Laboratory

³ Information Sciences Institute, University of Southern California

⁴ Department of Computer Science, University of Chicago

⁵ Lawrence Berkeley National Laboratory

⁶ Department of Computer Science, University of Wisconsin

1. Petascale Science is an End-to-end Problem

Petascale science is an end-to-end endeavor, involving not only the creation of massive datasets at supercomputers or experimental facilities, but the subsequent analysis of that data by a user community that may be distributed across many laboratories and universities. The new Center for Enabling Distributed Petascale Science (CEDPS) supported by the US Department of Energy's Scientific Discovery through Advanced Computing (SciDAC) program is developing tools to support this end-to-end process. In this brief article, we summarize the goals of the project and its progress to date. Some material is adapted from a longer article that appeared in the 2007 SciDAC conference proceedings [7].

At a recent workshop on computational science, the chair noted in his introductory remarks that if the speed of airplanes had increased by the same factor as computers over the last 50 years, namely five orders of magnitude, then we would be able to cross the US in less than a second. This analogy communicates with great effectiveness the remarkable impact of continued exponential growth in computational performance, which along with comparable improvements in solution methods is arguably the foundation for SciDAC.

However, a participant was heard to exclaim following these remarks: “*yes—but it would still take two hours to get downtown!*” The serious point that this speaker was making is that science is an end-to-end problem and that accelerating just one single aspect of the problem solving process can inevitably achieve only limited returns in terms of increased scientific productivity.

These concerns become particularly important as we enter the era of petascale science, by which we mean science involving numerical simulations performed on supercomputers capable of a petaflop/sec or higher performance, and/or experimental apparatus—such as the Large Hadron Collider [4], light sources and other user facilities [1], and ITER [3]—capable of producing petabytes of data. Successful science using such devices demands not only that we be able to construct and operate the simulation or experiment, but also that a distributed community of participants be able to access, analyze, and ultimately make sense of the resulting massive datasets. In the absence of appropriate solutions to the end-to-end problem, the utility of these unique apparatus can be severely compromised.

The following example illustrates issues that can arise in such contexts. A team at the University of Chicago recently used the FLASH3 code to perform the world's largest compressible homogeneous isotropic turbulence simulation [15]. Using 11 million CPU-hours on the LLNL BG/L computer over a period of a week, they produced a total of 154 terabytes of data, contained in 75 million files which subsequently were archived. Subsequently, they used GridFTP to move 23 terabytes of this data to computers at the University of Chicago; using four parallel streams, this took some three weeks at around 20 megabyte/sec. Next, they spent considerable time using local resources to tag the data, analyze, and visualize the data, augmenting the metadata as well. In a final step, they are making this unique dataset available for use by the community of turbulence researchers by providing analysis services so that other researchers can securely download portions of the data for their own use. In each

of these steps, they were ultimately successful—but they would be the first to argue that the effort required to achieve their end-to-end goals of scientific publications and publicly available datasets was excessive.

As this example illustrates, a complete solution to the end-to-end problem may require not only methods for parallel petascale simulation and high-performance parallel I/O (both handled by the FLASH3 code and associated parallel libraries), but also efficient and reliable methods for:

- high-speed reliable *data placement*, to transfer data from its site of creation to other locations for subsequent analysis;
- terascale or faster *local data analysis*, to enable exploration of data that has been fetched locally;
- high-performance *visualization*, to enable perusal of selected subsets and features of large datasets data prior to download;
- *troubleshooting* the complex end-to-end system, which due to its myriad hardware and software components can fail in a wide range of often hard-to-diagnose ways;
- building and operating *scalable services* [17], so that many users can request analyses of data without having to download large subsets [this aspect of the project is not addressed in this article];
- *securing* the end-to-end system, in a manner that prevents (and/or can detect) intrusions and other attacks, without preventing the high-performance data movement and collaborative access that is essential to petascale science; and
- *orchestrating* these various activities, so that they can be performed routinely and repeatedly.

Each of these requirements can be a significant challenge when working at the petascale. Thus, a new SciDAC Center for Enabling Technology, the Center for Enabling Distributed Petascale Science (CEDPS) was recently established to support the work of any SciDAC program that involves the creation, movement, and/or analysis of large amounts of data, with a focus on data placement, scalable services, and troubleshooting.

2. Current Data Placement Approaches

Large quantities of data must frequently be moved among computers in a petascale computing environment, whether because there are insufficient resources to perform analysis on the platform that generated the data, because analysis requires specialized resources or involves comparison with other data, or because the data must be published, that is, moved and augmented with metadata, to facilitate use by the community.

Our data placement work addresses three classes of application requirements. First, *staging to and from* active computations and workflows requires placement of data at advantageous locations. By using a data placement service to perform staging operations asynchronously with respect to a workflow or execution engine, rather than explicitly staging data at run time, we hope to demonstrate improved application performance, as suggested in simulations [20] and initial measurements of workflow execution [11]. A current example of where these methods can be applied is the visualization of the results of a combustion simulation at NERSC which produces 100 TB of data. Smarter placement of the data during simulation execution will enable better use of the visualization component and let scientists understand the resulting data in a more timely fashion.

Second, *archival storage* is often the final location of data products that are staged out of a running application, and better data placement services can make archiving operations more efficient. When an application runs on a compute resource such as a cluster or supercomputer, data products must often be staged off the storage system associated with that computational resource onto more permanent secondary or archival storage. These staging out operations can limit application performance, particularly if the compute resource is storage-limited; using an asynchronous data placement service to stage out data products should improve performance. For example, the team running the CCSM climate simulation code at ORNL wants to publish its output data to the Earth System Grid (ESG) [9].

They must both transfer the output data to an HPSS archive at NERSC (perhaps while the model is running) and also register each file in a metadata catalog for ESG.

Finally, we are interested in data placement services that maintain required levels of *redundancy* in a distributed environment. For example, it might be the policy of the data placement service to ensure that there are always three copies of every data item stored in the system. If the number of replicas of any data item falls below this threshold, the placement service is responsible for creating additional replicas to meet this requirement. An example of where this requirement arises in practice is the data produced by the CMS experiment at the LHC (at a sustained rate of 400 MB/s), which must be delivered to a Tier 1 site in the US for further processing and then distribution among several US domestic and 20 non-US Tier-2 sites.

Such scenarios, for which we can give many other examples across a wide range of applications, can involve many of the following six elements:

1. Data *registration* and metadata *tagging* as well as data movement;
2. *Bulk data transfer* over high-speed long-haul networks from different sources and sinks;
3. *Coordinated data movement* across multiple sources, destinations, and intermediate locations, including parallel file systems, virtual disks, and hierarchical storage, and among multiple users and applications;
4. *Failure reduction* techniques, such as storage reservation and data replication;
5. *Failure detection* techniques including online monitoring and operation retry to detect and recover from multiple failure modalities; and
6. A need for *predictability* and coordinated scheduling in spite of variations in load and competing use of storage space, bandwidth to the storage system, and network bandwidth.

To summarize the motivation for CEDPS in a sentence: *not only must we be able to transfer data and manage end-point storage systems and resource managers; we must also be able to support the coordinated orchestration of data across many community resources.*

Currently available tools address portions of this functionality. Basic high-performance data transfer (2) is supported by GridFTP [6], which provides fast performance through parallelism and stripping between data sources. The Replica Location Service and associated Globus data services [10] can provide basic ways to look up where a replica is stored, but metadata tagging (1) is generally an application-specific tool. The NeST [8] and dCache [21] storage management services provide disk-side support for data placement and some of the reliability and error prevention required (4), but not the broader coordinated data movement (3) needed by today's applications. Failure detection (5) and performance prediction (6) are considered open areas of research by many. In general, these requirements go well beyond our current data transfer and storage resource management capabilities. We will discuss the ways in which our new technology addresses these six elements in the following sections.

3. The CEDPS Managed Object Placement Service: MOPS

We are creating a new class of *data placement services* that can position data reliably across diverse systems and coordinate provisioning, movement, and registration across multiple storage systems to enable efficient and prioritized access by many users. A single logical transfer may involve multiple sources and destinations necessitating the use of intermediate store and forward storage systems or the creation of optimized overlay networks such as user level multicast networks. Concurrent independent placement operations may be prioritized and monitored in case of failures.

As a first step, we have recently released a prototype Managed Object Placement Service (MOPS), shown in Figure 1, which transforms storage into a managed resource. MOPS allows users to negotiate access to a certain quantity of storage for a certain time and with defined performance characteristics. Its design and implementation leverages GridFTP, NeST, and dCache.

GridFTP[8] provides a flexible core architecture with a data interface component that allows different plug-ins for added functionality. It is well known for its high-speed data transfer capabilities.

GridFTP gives MOPS the core functionality of fast, bulk file transfers, element 2 in our scenarios, which MOPS extends through its plug-in capability

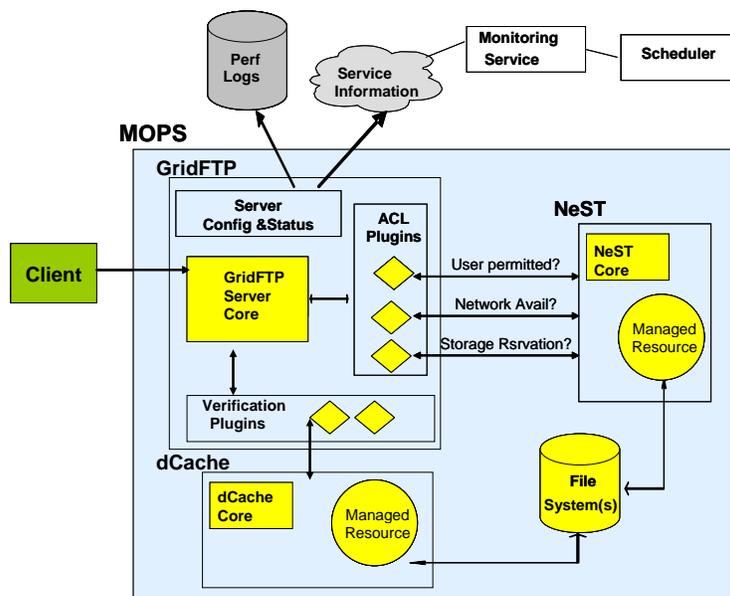


Figure 1: General MOPS architecture.

NeST [8] provides guaranteed storage allocation by allowing the user and storage device to negotiate a size and duration and to specify access control lists (ACLs) for file access. In this way a system can specify which users can access certain files or sets of files and also work with disk reservations when they are available. This feature helps address element 3, coordinated data movement, and element 4, failure reduction, by decreasing the chance of disk overflow errors.

dCache [22] provides methods for managing backend (tertiary) storage systems including space management, hot spot determination, and recovery from disk or node failures. When connected to a tertiary storage system, dCache simulates unlimited direct access storage space; data exchanges to and from the underlying tertiary storage system are performed automatically and invisibly to the user. Recent CEDPS-funded work has implemented data transfer consistency verification features for verifying that individual transfers have completed correctly. dCache also addresses element 3, coordinated data movement, and element 4, failure reduction.

By combining these three tools with a single user interface using MOPS, CEDPS users can now work with their data in a more managed environment, especially in terms of reducing failures due to running out of disk space in the middle of a transfer, limiting the access to a set of files, or verifying that a transfer has completed successfully, while continuing to serve the data quickly across a wide variety of networks and back-end storage systems.

4. The CEDPS Data Placement Service

CEDPS is also developing the Data Placement Service (DPS) that will perform data transfer operations using MOPS. For data-intensive scientific applications running in a distributed environment, the placement of data onto storage systems can have a significant impact on the performance of scientific computations and on the reliability and availability of data sets. These scientific applications may produce and consume terabytes or petabytes of data stored in millions of files or objects, and they may run complex computational workflows consisting of millions of interdependent tasks. A variety of data placement algorithms could be used, depending on the requirements of a scheduler or workflow management system as well as the data distribution goals of the scientific collaboration, or *Virtual Organization* (VO). For example, a placement algorithm might

distribute data in a way that is advantageous for application or workflow execution by placing data sets near high-performance computing resources so that they can be staged into computations efficiently; by moving data off computational resources quickly when computation is complete; and by replicating data sets for performance and reliability. These goals might be considered *policies* of the workflow manager or VO, and a *policy-driven data placement service* is responsible for replicating and distributing data items in conformance with these policies or preferences. A data placement service could also make use of hints from a workflow management system about applications and their access patterns, for example, whether a set of files is likely to be accessed together and therefore should be replicated together on storage systems.

To demonstrate the effectiveness of intelligent data placement, we integrated the Pegasus workflow management system [14] from USC Information Sciences Institute with the Globus Data Replication Service [13], which provides efficient replication and registration of data sets. We demonstrated [11] that using hints from the workflow management system allowed us to reduce the execution time of scientific workflows when we were able to successfully prestage necessary data onto appropriate computational resources.

This initial work has led us to design a general, asynchronous Data Placement Service (DPS) that will operate on behalf of a virtual organization and accept data placement requests from clients that reflect, for example, grouping of files, order of file requests, etc. Figure 2 illustrates the operation of a DPS for stage in requests issued by a workflow management system. We also plan to incorporate configurable policies into the data placement service that reflect the data distribution policies of a particular VO. Our goal is to produce a placement service that manages the competing demands of VO data distribution policies, data staging requests from multiple competing workflows, and additional on-demand data requests from other clients.

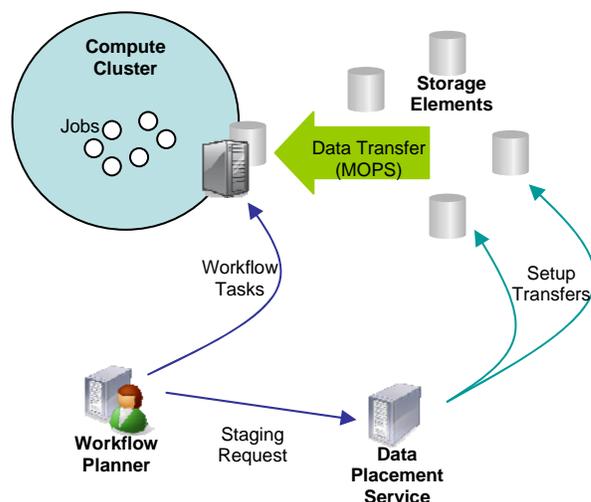


Figure 2: Shows a workflow management system acting as a client of a data placement service and issuing requests for staging in of data sets. The DPS issues MOPS data transfers from appropriate storage elements to the compute cluster(s) on which workflow execution will take place.

We have implemented an initial version of the data placement service with a planned software release in October 2007. This implementation modifies and significantly extends the existing Globus Data Replication Service. The implementation uses several Globus components, including the Java WS Core that provides the basic infrastructure for supporting web service deployment and generic operation support such as basic state management, query operations, endpoint references, etc.; the Globus Replica Location Service that provides registration and discovery of of data items; the GridFTP data transfer service for secure and efficient data staging operations; and the Grid Security Infrastructure for secure access to resources in the distributed environment.

5. CEDPS Troubleshooting

Distributed data management involves end-to-end systems comprising of many different hardware and software components in different physical locations and administrative domains. Failures can occur and they can be hard to diagnose. Experience with current DOE distributed system deployments has shown that understanding behavior is a fundamental requirement, not just a desirable enhancement. Middleware may also mask performance faults, when applications produce correct results but experience degradation in performance.

In order to better understand failures and to increase the reliability of the end-to-end system we have developed tools to allow easier access to logs and additional log analysis software that performs anomaly detection. In addition, we have also deployed a higher-level monitoring tool that observes services and generates notifications when errors occur.

Figure 3 shows the CEDPS log management service based on the syslog-ng system [5]. We mine software and service logs (such as those from GridFTP, MOPS, or other tools) which are filtered and forwarded to a common location. That combined set of data can then be analyzed. We have used NetLogger [22] to access performance data and discover faulty event chains where expected behavior does not occur. We have also developed prototypes of anomaly detection tools that can detect a missing event in an event stream and also identify unexpected performance variations which indicate an underlying problem that may not cause an outright failure [12]. This system is currently in the process of being deployed on the Open Science Grid (OSG) [19].

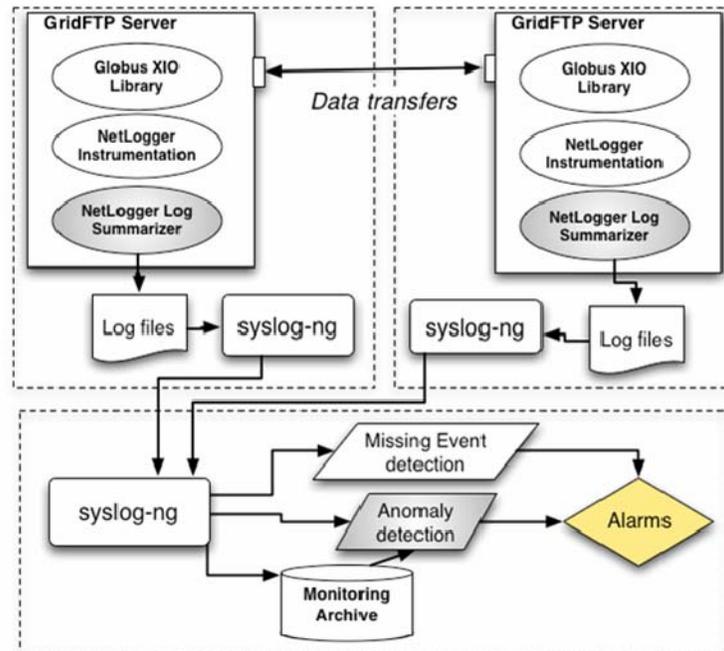


Figure 3: Syslog-ng deployment architecture, and interactions with anomaly detection and alarm tools.

Both of these tools have been aided by effort spent on improving the quality and consistency of available performance information. Specifically, we have codified a set of logging “Best Practices” [2], and are modifying the Globus Toolkit [16] to follow these practices. In defining these guidelines, we have worked with the European EGEE project to achieve compatibility with their security logging guidelines [18], an important requirement for LHC computing.

To compliment our log services and to assist further with our scenario elements 5 and 6, failure reduction and detection, we have also developed a Trigger service [12] that runs small probes and notifies system administrators and end users when certain conditions are met. These can include a service failure or failure to respond to a ping, or a warning condition, such as a nearly full disk, overly

long queue, or high load condition on a resource. The Trigger service has been used by ESG for over three years for system failure notifications and to help diagnose errors. We have re-architected this component to allow for additional trigger services, a separation of matching conditions and actions taken upon failure notification, and easier deployment through a Web interface.

These tools combine to give us additional support in the end-to-end data management environment.

6. Revisiting the FLASH Example

We began this article with a discussion of the University of Chicago FLASH application experiment, in which it took three weeks at 20 MB/s to transfer less than 15% of the data produced in a three-week simulation. By using MOPS, it is possible that smarter disk allocation could have been done, allowing the FLASH group to transfer data of particular interest more quickly and as it was being generated due to smarter handling of the backend storage system. When performing local analysis and replication of the data, the FLASH team could now take advantage of the DPS which would handle registering new files and distributing them according to the policy defined by the FLASH team, instead of having to do this work by hand. In addition, with the added centralized logging and trigger service deployed at the various sites, FLASH scientists would be able to detect any failures and debug any performance problems much more easily than the current environment. The effort required to achieve their end-to-end goals of scientific publications and publicly available datasets would be significantly reduced overall.

7. Summary

We have introduced the SciDAC Center for Enabling Distributed Petascale Science (CEDPS), which is addressing three problems critical to enabling the distributed management and analysis of petascale datasets: data placement, scalable services, and troubleshooting.

In data placement, we are developing tools and techniques for reliable, high-performance, secure, and policy driven placement of data within a distributed science environment. We are constructing a managed object placement service (MOPS)—a significant enhancement to today's GridFTP—that allows for management of the space, bandwidth, connections, and other resources needed to transfer data to and/or from a storage system. Building on this base, we are developing end-to-end data placement services that implement different data distribution and replication behaviors.

In troubleshooting, we are developing tools for the *detection and diagnosis of failures* in end-to-end data placement and distributed application hosting configurations. We are constructing an end-to-end monitoring architecture that uses instrumented services to provide detailed data for both background collection and run-time event-driven collection. We are also constructing new monitoring analysis tools able to detect failures and performance anomalies and predict system behaviors using archived data and event logs.

These tools allow scientists to interact more easily with large data sets created during petascale computations, and allow faster end analysis of the data. More details can be found at www.cedps.net.

Acknowledgements

This work is supported through the U.S. Department of Energy Office of Science, Office of Advanced Scientific Computing Research, through the SciDAC program. Work at Argonne is supported under Contract DE-AC02-06CH11357 and at Lawrence Berkeley National Laboratory, under Contract DE-AC02-05CH11231. We gratefully acknowledge the contributions of our fellow CEDPS participants Andrew Baranovski, Shishir Bharathi, John Bresnahan, Tim Freeman, Keith Jackson, Kate Keahey, Carl Kesselman, David E. Konerding, Mike Link, Miron Livny, Neill Miller, Robert Miller, Gene Oleynik, Laura Pearlman, and Robert Schuler.

References

1. BES Scientific User Facilities, <http://www.sc.doe.gov/bes/BESfacilities.htm>, 2007.

2. Grid Logging Best Practices Guide, CEDPS, <http://www.cedps.net/wiki/images/6/6f/CEDPS-troubleshooting-bestPractices-16.doc>, 2007.
3. ITER, www.iter.org, 2006.
4. LHC - The Large Hadron Collider Project, <http://lhc.web.cern.ch>, 2007.
5. The syslog-ng Logging System, <http://www.balabit.com/products/syslog-ng/>, 2007.
6. Allcock, B., Bresnahan, J., Kettimuthu, R., Link, M., Dumitrescu, C., Raicu, I. and Foster, I. The Globus Striped GridFTP Framework and Server *SC'2005*, 2005.
7. Baranovski, A., et al. Enabling Distributed Petascale Science. *Journal of Physics: Conference Series*, 78. 2007.
8. Bent, J., et al. NeST: A Grid Enabled Storage Appliance *Grid Resource Management: State of the Art and Future Trends*, 2004.
9. Bernholdt, D., et al. The Earth System Grid: Supporting the Next Generation of Climate Modeling Research. *Proceedings of the IEEE*, 93 (3). 485-495. 2005.
10. Chervenak, A., et al. Giggle: A Framework for Constructing Scalable Replica Location Services *SC'02: High Performance Networking and Computing*, <http://www.globus.org/research/papers.html#giggle>, 2002.
11. Chervenak, A., et al. Data Placement for Scientific Applications in Distributed Environments *8th IEEE/ACM International Conference on Grid Computing (Grid 2007)*, Austin, TX, 2007.
12. Chervenak, A., et al. Monitoring the Earth System Grid with MDS4 *2nd IEEE Intl. Conference on e-Science and Grid Computing (e-Science 2006)*, Amsterdam, Netherlands, 2006.
13. Chervenak, A., Schuler, R., Kesselman, C., Koranda, S. and Moe, B., Wide Area Data Replication for Scientific Collaborations. in *6th IEEE/ACM Int'l Workshop on Grid Computing (2005)*.
14. Deelman, E., et al. Pegasus: A Framework for Mapping Complex Scientific Workflows onto Distributed Systems. *Scientific Programming*, 13 (3). 219-237. 2005.
15. Fisher, R.T., et al. Terascale Turbulence Computation on BG/L Using the FLASH3 Code. *IBM Systems Journal*. 2007.
16. Foster, I. Globus Toolkit Version 4: Software for Service-Oriented Systems. *Journal of Computational Science and Technology*, 21 (4). 523-530. 2006.
17. Foster, I. Service-Oriented Science. *Science*, 308. 814-817. 2005.
18. Groep, D. *Middleware Security Audit Logging Guidelines* EGEE Document 2006-11-07, <https://edms.cern.ch/document/793208>, 2006.
19. Pordes, R., et al., The Open Science Grid. in *Scientific Discovery through Advanced Computing (SciDAC) Conference*, (2007).
20. Ranganathan, K. and Foster, I. Simulation Studies of Computation and Data Scheduling Algorithms for Data Grids. *Journal of Grid Computing*, 1 (1). 2003.
21. Shoshani, A., Sim, A. and Gu, J. Storage Resource Managers: Essential Components for the Grid. in Nabrzyski, J., Schopf, J.M. and Weglarz, J. eds. *Grid Resource Management: State of the Art and Future Trends*, Kluwer Academic Publishers, 2003.
22. Tierney, B. and Gunter, D. *NetLogger: A Toolkit for Distributed System Performance Tuning and Debugging*. Lawrence Berkeley National Laboratory, Technical Report LBNL-51276, 2003.