GridMapper: A Tool for Visualizing the Behavior of Large-Scale Distributed Systems

William Allcock Joseph Bester John Bresnahan Ian Foster Jarek Gawor
Joseph A. Insley Joseph M. Link Michael E. Papka
Mathematics and Computer Science Division, Argonne National Laboratory,
Argonne, IL 60439, USA
{insley, papka}@mcs.anl.gov

Abstract

Grid applications can combine the use of compute, storage, network, and other resources. These resources are often geographically distributed, adding to application complexity and thus the difficulty of understanding application performance. We present GridMapper, a tool for monitoring and visualizing the behavior of such distributed systems. GridMapper builds on basic mechanisms for registering, discovering, and accessing performance information sources, as well as for mapping from domain names to physical locations. The visualization system itself then supports the automatic layout of distributed sets of such sources and animation of their activities. We use a set of examples to illustrate how the system can provide valuable insights into the behavior and performance of a range of different applications.

1. Introduction

A growing number of applications use collections of geographically distributed resources, with scope and membership changing over time. This increased complexity makes it important to understand interactions among the various components of a distributed system.

One way of gaining insight into the behavior of such systems is to visualize their activities. Seeing when and where jobs are executed and how data is transferred among jobs can help users discover performance bottlenecks and diagnose erroneous behavior. It also enables one to monitor the progress of a computation and to observe the overall state of the system. Visualizing network connectivity among the distributed resources on which distributed applications execute can reap similar benefits. Such visualizations can also serve as a valuable demonstration tool, making it easier to explain complex systems to someone unfamiliar with them.

We present here GridMapper, a tool that integrates application-specific performance data with network routing and connectivity data to create interactive, quasi-real-time visualizations of Grid applications that communicate, among other things, the geographical location of application components and the resources that they use, and the nature of the interactions among these components and resources.

We present the GridMapper architecture, explain how it integrates with application monitoring systems and other system components, and use three examples to illustrate the capabilities of our GridMapper prototype. Our experience with these applications persuades us that interactive, dynamic display and geographical location information can be a powerful combination for understanding (and communicating) Grid application behavior and performance.

2. Background and Related Work

We introduce the technologies that GridMapper uses to obtain geographic location information, and also review some previous work on the monitoring and visualization of distributed systems.

One obstacle to the accurate depiction of the location of distributed resources is obtaining accurate location information [1]. IPtoLL was an early tool for converting hostnames to latitude/longitude, but its highest degree of accuracy is based on the city [2]. More recently, the Cooperative Association for Internet Data Analysis (CAIDA) [3] has developed a tool called NetGeo that can be used to map IP addresses, domain names, and Autonomous System (AS) numbers to geographic locations [4]. While this tool has proved useful, the data it produces often has inaccuracies, mainly because of the granularity of the information that is available.

GTrace [5], also from CAIDA, is a graphical traceroute tool that uses NetGeo as one of its methods of determining geographic location. It plots the locations of each node along the path of the traceroute on a map and gives pertinent information about each, including the method used to determine the location and its level of trust in the results. It also displays routes that were traced

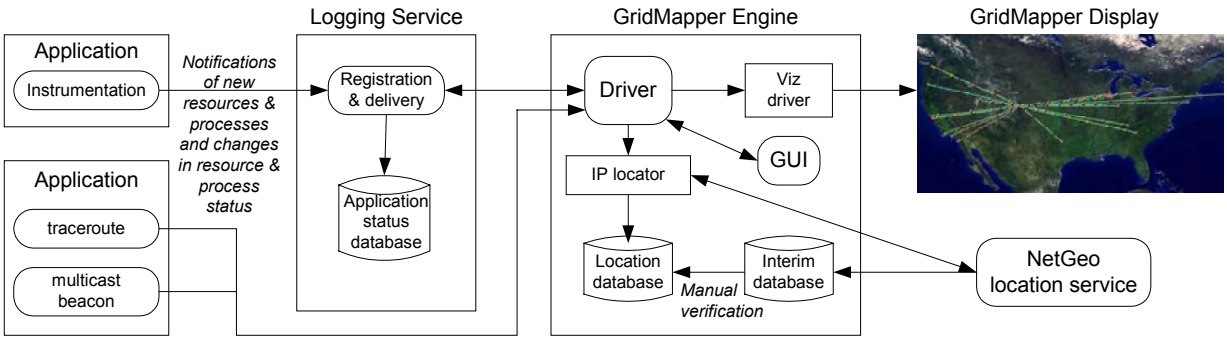

Figure 1. Overview of GridMapper architecture. The GridMapper engine contains a service for determining geographical locations (IP locator), and can obtain performance data either from instrumented applications, through the use of a Logging Service, or directly from systems that provide network characteristics.

via specialized third-party traceroute servers. GTrace does an excellent job of showing the path that data takes as it moves through the network, but alone it does not offer the range of information that we want to provide.

The NLANR Multicast Beacon [6] is a set of software components for monitoring the performance of multicast transmissions on a network. The Beacon Client runs on a set of machines that continuously send packets to each other through a multicast session. It measures the performance of the transmission and periodically reports those measurements to a central Beacon Server. Measurements include the percentage of packet loss from one client to another, one-way delay from one client to another, variation of the one-way delay, percentage of packets that arrived out of order, and percentage of duplicate packets. This data is generally displayed one variable at a time in a simple N x N matrix. These displays do not provide any information about the geographical locations of the various hosts.

Other tools for visualizing general network performance [7, 8, 9] show the number of bytes moving across particular links in a network.

Each of these tools offers useful but independent capabilities. Our GridMapper visualization system integrates several of these capabilities to enable the visualization and monitoring, not only of network traffic for specific applications, but also of the activity of compute and other resources associated with those applications. Moreover, it provides routing and latency information, via traceroute data, and visualizes multicast connectivity and performance, using data from the multicast beacon.

3. Visualization System

The GridMapper visualization system consists of a flat map of the world on which are placed 3D primitives representing resources and network connections between those resources. Data transfers are represented by spheres that move along network connections. The visualization system's interactive interface allows the user to zoom in and out, pan across the map, and select resources to obtain more detailed information. A graphical user interface (GUI) provides additional controls and more textual information. An overview of GridMapper's components is shown in Figure 1.

The application runs on either a desktop workstation or a tiled display (Figure 2). Display clutter has been identified as a serious problem when displaying large networks [10]. The advantage of using the tiled display is that the large format and increased resolution enable a higher level of detail of the map, thus reducing clutter.

3.1. Resource Placement

We wish to obtain latitude/longitude information for arbitrary IP addresses (resources) in order to provide a geographically realistic depiction of resource location. Because existing information sources cannot be counted upon to be accurate, we adopt a tiered approach based on a central, manually verified database of definitive locations backed up by CAIDA's NetGeo service.

Each resource entry in the central database contains information such as site name, domain, hostname, IP address, city, state, country, latitude, and longitude. When trying to determine a resource's location, we first look in this database. (The database is small enough to maintain in memory, so lookups are fast.) If the resource is not found there, we consult NetGeo.

Information retrieved from NetGeo is both used for visualization and stored in a temporary database for manual verification before being moved into the permanent database. In the future, we may transform our database into a separate service. We can also imagine obtaining location information from the resources themselves, if they choose to publish it via a system such

Figure 2. GridMapper's visualization system running on Argonne's μMural2 six-projector tiled display.

as the Globus Toolkit™'s Monitoring and Discovery Service (MDS) [11].

3.2. Traceroute

We next address the problems of how to determine and represent properties of the network links that connect resources. A variety of systems exist for determining the physical topology of network connections (e.g., traceroute [12]) and both end-to-end and link-by-link characteristics (e.g., Network Weather Service [13], pipechar [14]). In our initial work, we use traceroute to obtain both topology and latency information; as we explain below, end-to-end information can be obtained from applications if desired.

We run traceroute from one host to another and then visualize the route that is returned, by placing a cone on the map where each router in the trace is located and connecting these cones with tubes to represent the links between hops. Links are color-coded based on the latency for their trace segment. Spheres move along the links for each segment at varying speeds; the time to traverse a link is proportional to the latency of that link (Figure 3).

3.3. Grid Monitoring

One obstacle to visualizing the progress and performance of actual applications is obtaining appropriate data from these applications. This task will, we hope, be made easier in the future by the adoption of uniform monitoring and discovery mechanisms, such as those defined by the Open Grid Services Architecture (OGSA) [15] and discussed within the Grid Monitoring Architecture group at the Global Grid Forum [16].

In the meantime, we have developed our own custom event infrastructure for use in our work. This infrastructure comprises a logging server to which applications can send events, plus APIs for sending events to and receiving events from this server. The service both archives events and forwards them to any clients that have asked to receive them. Hence, events received from

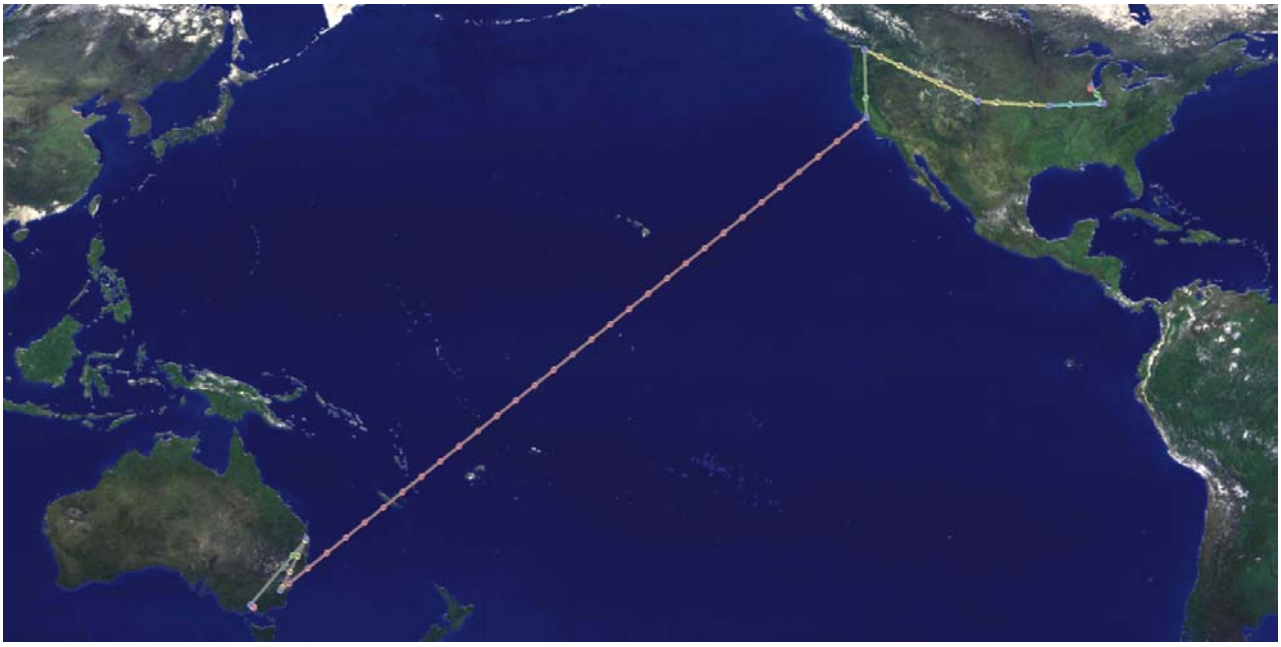

Figure 3. GridMapper display showing the path obtained by running traceroute from a host at Argonne National Laboratory in Argonne, Illinois, to Monash University in Clayton, Australia.

applications can be visualized in real time, at a later time, or both.

Our prototype service does not address security issues, which would of course be required in a production implementation. We expect the development of OGSA to lead to the development of more sophisticated services that address these issues.

3.3.1. Logging Capabilities. We base our logging server and APIs on the NetLogger toolkit, a set of tools that make it easy for distributed applications to log interesting events [17, 18].

Log messages are text strings structured as a list of "field=value" tuples with, in our case, required fields including DATE, HOST, PROG (program), LVL (level), TYPE, ID (unique identifier), and FRIENDLY.NAME (human-readable string). The last two fields are used to identify particular streams of events when presented with a list of streams. The TYPE field distinguishes among different event types and thus dictates what other tuples should be present in the string. In particular, we define a *transfer type*, which gives performance measurements associated with data transfers, and a *job type*, which indicates the number (and state) of jobs running on a compute resource.

In order to obtain useful information based on these events, it is critical that the clocks of all systems be synchronized. We accomplish this synchronization by using NTP [19].

3.3.2. Logging Server. Our server extends the capabilities of *netlogd*, a simple program provided with the NetLogger toolkit that reads data from the network and writes it to a file. The various components of a distributed system that we wish to monitor with GridMapper are then instrumented to send performance data and other events to this server.

Our server can log events from many streams simultaneously, putting all events with the same stream ID field into its own text file. Hence, events from different providers, potentially logging different types of events, can be treated as a single stream of events.

Clients—in the case of GridMapper, our visualization application—can use the client API to make queries to the server and subscribe to particular event streams.

3.3.3. Logging Client API. The logging client API enables three important interactions with the server:

- *Queries*: Users can request a list of all event streams that meet criteria expressed as a set of required "field=value" pairs. Users can also filter requests based on whether any providers of a stream are currently connected to the server (i.e., if there is live data). The user provides a callback function for processing list entries, which gets called for each entry in the list.
- *Subscriptions*: Users can subscribe to, and unsubscribe from, streams based on the stream ID. When subscribing, they can specify the time range of the events that they are interested in receiving. A

time range can encompass both archived (past) and live (current and future) events. Once subscribed to a stream, the client again gets callbacks, to a function that the user provides, for each event in the stream. Unsubscribing simply means closing the connection with the server and no longer receiving events.

- *Notify*: Users can subscribe to a special "notify" stream that is produced by the server. This stream sends a begin event after the first message it receives for a new stream ID and an end event after the last connection sending events for that stream ID has been closed. Users can specify filters to be used when subscribing to the notify stream, so that they are notified only of the particular streams in which they are interested.

The API also provides convenience methods that users can use in their callback functions for parsing values out of the stream lists as well as out of the events.

3.3.4. Visualization Control: Listing and Subscribing. The GUI of our visualization tool provides a number of controls for Grid monitoring. There is a place to enter the hostname and port where the logging server is listening. There is a region for managing list requests, which allows the user to specify filters to use in those requests. This region is also where request results are displayed, along with buttons for subscribing to and unsubscribing from the streams.

Another region supports the handling of the notify stream. Here the user can specify filters to use on the notify stream, see the list of streams of which they have been notified, and subscribe to and unsubscribe from those streams.

There are also buttons that allow the user to automatically subscribe to streams when they begin sending events and/or unsubscribe from them when they stop sending events.

Another group of controls on the GUI deals with data that is currently being visualized. There is a list here of all the streams that are currently subscribed to, with a button for unsubscribing. It is necessary to have this list in addition to having these items in the List or Notify regions because the lists in the other two regions are subject to change and it is possible for a subscribed stream to be removed from those lists, leaving the user without a way to unsubscribe. This region also contains widgets for controlling different aspects of the actual visualization of the data (described below).

3.3.5. Visualizing the Data. When the visualization application receives the first transfer event for a stream, it plots the endpoints of the transfer on the map, adds a link between the two sites, and stores the statistics for the transfer (total number of bytes transferred so far, current bandwidth, and average bandwidth). For subsequent events for that transfer, the visualization tool locates the appropriate sites and link and updates their values. Spheres are moved along the link in the direction of the data flow. Each sphere represents a particular amount of

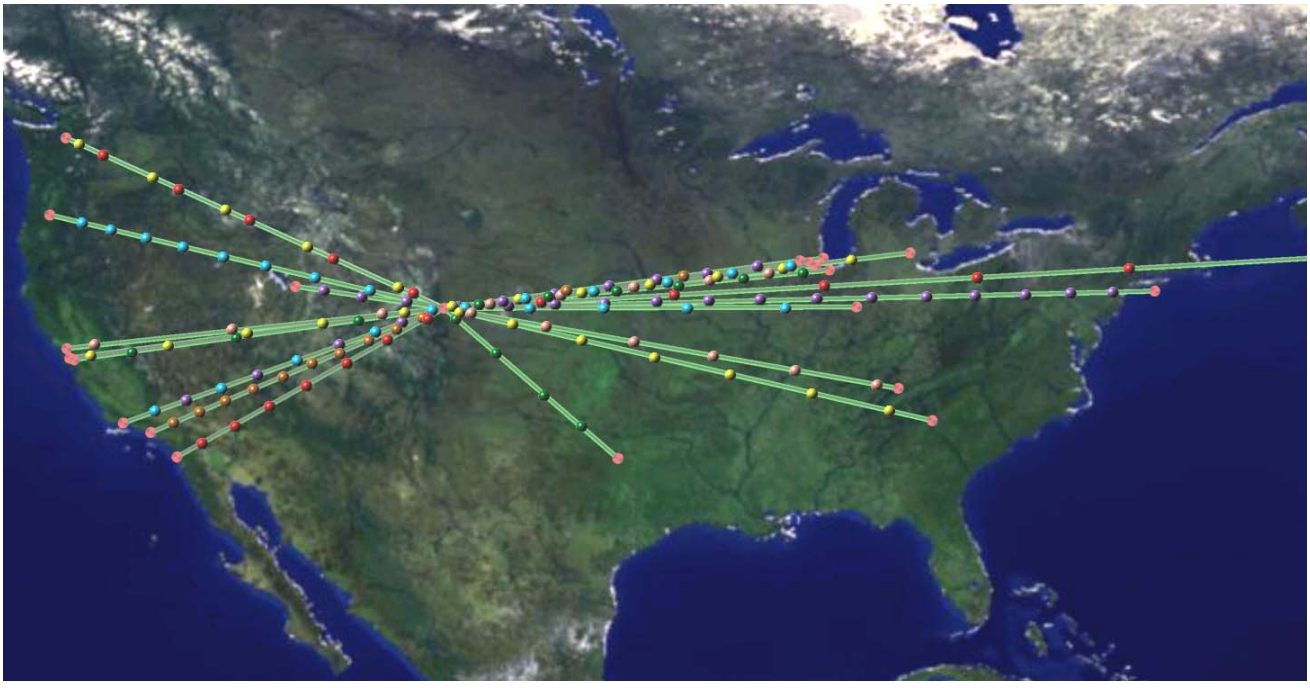

Figure 4. Numerous transfers from multiple hosts around the United States, and one in Europe, all moving data through the GridFTP proxy server in Denver, Colorado during a demonstration at SC2001.

data, between five and one hundred megabits, which is controlled from a slider on the GUI, and moves at a constant speed along the link, also controlled from the GUI. Additional spheres are placed onto the link based on the current bandwidth. When multiple transfers are moving in the same direction along a link, the user can chose to display a different set of spheres for each transfer (each set a different color) or a single set that represents the total bandwidth of all transfers.

For job events, the tool plots the site where the jobs are running and keeps track of the number of jobs submitted, completed successfully, and failed. The site is highlighted while jobs are actively running there.

Selecting a site on the map brings up a text region with information about the site, hosts who have jobs that are being monitored, and transfers into and out of this site.

4. Application Examples

We use three examples to illustrate GridMapper's capabilities: a data transfer system, a remote computing application, and a collaborative environment.

4.1. GridFTP

We exploited logging capabilities incorporated in the GridFTP data transfer tools [20, 21] to enable visualization of wide area data transfers. In a demonstration at the SC2001 conference we visualized the activities of a prototype GridFTP proxy server [22]. With a GridFTP server running on each node of two eight-node clusters, the front-end proxy server receives all initial connections and forwards transfer requests to the backend server with the lightest load. We used our visualization tool to monitor 58 hosts spread across 21 sites as they transferred data to the proxy server, at a sustained rate of ~1.8 gigabits/sec and peak rate of ~2.8 gigabits/sec (Figure 4).

4.2. Remote Computing Application

We have used GridMapper to monitor a prototype remote computing application developed within the National Fusion Collaboratory project [23] (Figure 5). In this application, data is transferred from a storage location (on the left side of the image) to several compute resources that perform some computation (on the right side of the image). The results are then transferred back to the storage resource. The data is then transferred to another resource (which happens to be co-located with the storage system) where a visualization client displays it.

4.3. Access Grid Beacon

We extended our tool to support visualization of NLANR Multicast Beacon data. The raw data from the beacon is made available from a URL, which we access using capabilities provided by the Globus Toolkit [24].

Figure 5. This image shows a view of a run of the National Fusion Collaboratory application. The inverted cones above the two sites indicate that active jobs are currently running at those sites.

We again place cones on the map to represent sites where Beacon Clients are being run. Since multiple beacons can be running from a single location, all beacons originating from the same domain are listed under a single site. Selecting a site on the map brings up a text display with information about that site: site name, domain, city, state, country, and details about each beacon at that site: index (as assigned to it by the Beacon Server), hostname, and IP address.

Links are drawn from each site to all other sites and are color-coded based on the value of the parameter being displayed: loss, delay, jitter, order, or duplicate. The particular values that the colors represent differ slightly between parameters, but in general they are green (good), yellow (fair), red (poor), and gray (data either stale or unavailable). The user can select which parameter to display on the links. A legend shows the values represented by each color for the current parameter.

The GUI displays a list of all of the beacons, ordered by index, giving the site name and hostname of each beacon. The map can quickly become cluttered and unreadable if all links are shown at once. Therefore, the GUI has controls for selecting which links to display. The user can easily show only those links that have valid data or, conversely, only those with stale data. This feature can be useful when trying to identify sites in the multicast session that are having connectivity problems. The user can also toggle on/off all links into/out of individual sites.

The Access Grid (AG) is a system for distributed group-to-group collaboration [25, 26], including communication through audio and video streams broadcast via multicast. The AG community uses the NLANR multicast beacon to monitor connectivity between AG nodes. In addition to network measurements, the AG beacon contains information about the hosts running the beacon, such as the architecture of the machine and versions of the software. (Again, this information could in the future be provided in a more uniform fashion via a Grid information service such as MDS [11].) We have used our visualization tool to successfully monitor all of this AG beacon data. The additional information is made available on the GUI of our tool. Using our tool to visualize this data not only gives insight into the connectivity issues of the AG but also helps illustrate the extent to which the AG has grown by showing the distribution of sites around the world (Figure 6).

5. Future Work

Our GridMapper prototype uses a combination of our own custom notification mechanisms and NetLogger-based services to obtain performance information from applications. In the future, we will explore the use of notification and discovery mechanisms provided within the Open Grid Services Architecture both for registering

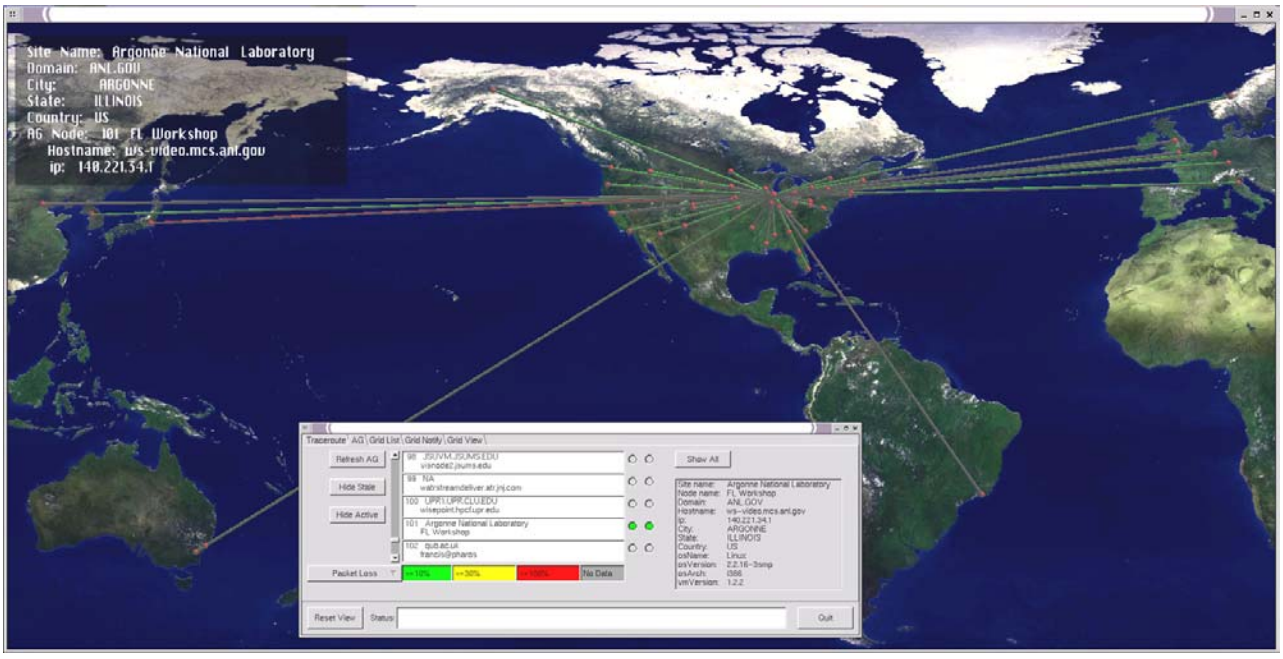

Figure 6. The visualization of the AG multicast beacon shows in this view all of the links into and out of one particular beacon. In the upper left, we show the textbox that appears when a site is highlighted. The GUI used to control the application, shown at the bottom, is actually a separate window, which has been placed on top of the graphics window in this view.

arbitrary "services" (e.g., application process or resource) with GridMapper and for obtaining information from those services.

We have also identified several directions to pursue with our visualization tool. Of foremost concern is improving upon the methods used to visualize the data. We are exploring different techniques for representing different types of data. This will become increasingly important as we continue to include more information from additional data sources. We will also explore the scalability of the system in terms of the number of data sources that can be processed and displayed.

Also, continued enhancement of current capabilities is important. In particular, we need to find new and better ways of obtaining accurate positioning information. We would like to expand on the traceroute capabilities, for instance adding the ability to do third-party traceroutes. We are considering using Globus Toolkit services for resource allocation and remote job submission to accomplish this. We could then show the actual paths that data transfers take, rather than showing that information out of band. Similarly, we would like to show results from mtrace, a tool for doing multicast traceroutes, and other multicast information. We also plan to add filtering to the multicast beacon visualization. This will allow users to make requests such as "Show me all of the links that have a loss rate of greater than twenty percent."

As the applications that we monitor are dynamically changing, it would be helpful to be able to see not only the current state but also how performance has changed over time. We plan to add graphing capabilities to help illustrate these changes. Another capability that we would like to provide is access to our visualization via the Web. The current implementation requires significant graphics power, however, so a Web interface will require different methods of visualization.

6. Conclusion

Visualizing the performance and interactions of dynamically changing resources in a distributed computing environment is a complex task. Getting accurate information about the location of resources is particularly difficult. Getting performance data from applications can also be a challenge, especially without making changes to the code itself. Building applications on top of tools that have instrumentation built into them can greatly simplify this process.

Using visualization systems such as the one presented here offers the immediate benefit of enabling one to see things happening, confirming that the system is working (or not working). When failure or bottlenecks occur, visualization can help identify the causes, or at least the location of the problem.

Much work remains to be done, but we believe that the tools that we have developed have brought us a step closer to understanding the benefits of distributed systems visualization. Further investigation is required to reveal its ultimate usefulness.

Acknowledgments

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Advanced Scientific Computing Research, U.S. Department of Energy, under Contract W-31-109-Eng-38 and included partial support under the auspices of the Scientific Discovery through Advanced Computing Initiative.

References

[1] V. N. Padmanabhan and L. Subramanian, An Investigation of Geographic Mapping Techniques for Internet Hosts. *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications.* ACM Press, New York, 2001, pp. 173-185.

[2] R. Olson, IPtoLL. *http://wwwunix.mcs.anl.gov/~olson/IPtoLL.html*

[3] CAIDA, the Cooperative Association for Internet Data Analysis. *http://www.caida.org*

[4] D. Moore, R. Periakaruppan, J. Donohoe, and K. Claffy, Where in the World Is netgeo.caida.org? *http://www.caida.org/outreach/papers/2000/inet_netgeo/*

[5] R. Periakaruppan and E. Nemeth, GTrace – A Graphical Traceroute Tool. *Proceedings of 13th Systems Administration Conference - LISA 1999.*

[6] NLANR Multicast Beacon. *http://dast.nlanr.net/Projects/Beacon/*

[7] Multi Router Traffic Grapher. *http://people.ee.ethz.ch/~oetiker/webtools/mrtg/*

[8] VisualRoute. *http://www.visualware.com/visualroute/index.html*

[9] Viznet. *http://dast.nlanr.net/projects/viznet/doc/*

[10] S. G. Eick, Aspects of Network Visualization. *IEEE Computer Graphics and Applications,* March 1996, pp. 69-72.

[11] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman, Grid Information Services for Distributed Resource Sharing. *Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10)*, IEEE Press, August 2001.

[12] V. Jacobson, Traceroute source code and documentation, 1988. Available from: ftp://ftp.ee.lbl.gov/traceroute.tar.gz.

[13] R. Wolski, N. Spring, and J. Hayes, The Network Weather Service: A Distributed Resource Performance Forecasting Service for Metacomputing, *Journal of Future Generation Computing Systems*, Volume 15, Numbers 5-6, pp. 757-768, October, 1999.

[14] G. Jin, G. Yang, B. Crowley, D. Agarwal, Network Characterization Service (NCS), *Proceedings of the 10th IEEE Symposium on High Performance Distributed Computing (HPDC-10)*, IEEE Press, August 2001.

[15] I. Foster, C. Kesselman, J. Nick, and S. Tuecke, The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. *http://www.globus.org/research/papers/ogsa.pdf* January, 2002.

[16] B. Tierney, R. Aydt, D. Gunter, W. Smith, V. Taylor, R. Wolski, and M. Swany, A Grid Monitoring Architecture. *http://www-didc.lbl.gov/GGF-PERF/GMA-WG/papers/GWD-GP-16-2.pdf*

[17] D. Gunter, B. Tierney, B. Crowley, M. Holding, and J. Lee, NetLogger: A Toolkit for Distributed System Performance Analysis. *Proceedings of the IEEE Mascots 2000 Conference*, August 2000, LBNL-46269.

[18] B. Tierney, W. Johnston, B. Crowley, G. Hoo, C. Brooks, and D. Gunter, The NetLogger Methodology for High Performance Distributed Systems Performance Analysis. *Proceedings of IEEE High Performance Distributed Computing Conference*, July 1998, LBNL-42611.

[19] D.L. Mills, Adaptive Hybrid Clock Discipline Algorithm for the Network Time Protocol. *IEEE/ACM Trans. Networking 6, 5*, October 1998, pp. 505-514.

[20] W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke, Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing**.** *IEEE Mass Storage Conference*, 2001.

[21] W. Allcock, A. Chervenak, I. Foster, C. Kesselman, C. Salisbury, and S. Tuecke, The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets**.** *Journal of Network and Computer Applications*, 23:187-200, 2001.

[22] SC2001 Bandwidth Challenge Entry, Simulation of a High Energy Physics (HEP) Tier One Site, *http://www-fp.mcs.anl.gov/dsl/scidac/datagrid/sc2001_bw_challenge_entry.htm*

[23] K. Keahey, T. Fredian, Q. Peng, D. P. Schissel, M. Thompson, I. Foster, M. Greenwald, and D. McCune, Computational Grids in Action: The National Fusion Collaboratory, accepted for publication in the *Journal of Future Generation Computer Systems*.

[24] I. Foster, C. Kesselman, Globus: A Toolkit-Based Grid Architecture. In I. Foster and C. Kesselman eds. *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, 1999, 259-278; and http://www.globus.org.

[25] L. Childers, T. Disz, R. Olson, M. E. Papka, R. Stevens, and T. Udeshi, Access Grid: Immersive Group-to-Group Collaborative Visualization, *Proceedings of the Fourth International Immersive Projection Technology Worksho*p, June 19-20, 2000

[26] L. Childers, T. L. Disz, M. Hereld, R. Hudson, I. Judson, R. Olson, M. E. Papka, J. Paris, and R. Stevens, ActiveSpaces on the Grid: The Construction of Advanced Visualization and Interaction Environments, *Parallelldatorcentrum Kungl Tekniska Högskolan Seventh Annual Conference (Simulation and Visualization on the Grid),* vol. 13, Lecture Notes in Computational Science and Engineering, B. Engquist, L. Johnsson, M. Hammill, and F. Short, Eds. Stockholm, Sweden: Springer-Verlag, 1999, pp. 64-80