WebViz: A Tool for WWW Access Log Analysis

James E. Pitkow & Krishna A. Bharat

Graphics, Visualization and Usability Center
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332-0280
E-mail {pitkow, kb}@cc.gatech.edu

ABSTRACT

Various programs have emerged that provide statistics analysis of World Wide Web (WWW) access logs. These programs typically detail the number of accesses for a file, the number of times a site has visited the database, and some programs even provide temporal analysis of requests[1]. However, these programs are not interactive nor do they provide visualizations of the local database. WebViz was developed with the intention of providing WWW database maintainers and designers with a graphical view of their local database and access patterns. That is, by incorporating the Web-Path paradigm into interactive software, users can see not only the documents (represented visually as nodes) in their database but also the hyperlinks travelled (represented visually as links) by users requesting documents from the database. WebViz further enables uses to selectively filter the access log (e.g. restrict the graphical view by specifying the desired domain names or DSN numbers, directory names, and start and stop times), control bindings to graph attributes (e.g. node size, border width and color as well as link width and color can be bound to frequency and recency information), play back the events in the access log (e.g. re-issue the logged sequence of requests), select a layout of nodes and links that best presents the database's structure, and examine the graph at any instant in time. Clearly, WebViz is a useful WWW database utility given that it can provide the user with graphical information about document accesses and the paths taken by users through the database. Such analyses can facilitate structural and contextual changes resulting in a more efficient use of the document space. This paper details the implementation of WebViz and outlines possible future extensions.

INTRODUCTION

WWW database developers, designers, and maintainers have a potentially formable task in analyzing the overall efficiency of their database. Following in the footsteps of the all-too- common end-user question: "Where am I?" [Nielson, 1990], comes the database-provider question: "How are people using our database?" The latter question requires analyses of the structure of the hyperlinks as well as the content of the documents in the database. The end products of such analyses might include 1) the frequency of visits per document, 2) the most recent visit per documents, 3) who is visiting which documents, 4) the frequency of use of each hyperlink, and 5) the most recent use of each hyperlink. Granted, this list does not include all potentially useful analyses; rather, it provides a starting point for the development of tools to provide such functionality. Towards this end, we developed a C++ visualization tool (running on SunOS 4.1.3 and X) called WebViz. The next section describes the underlying concept of WebViz, the Web-Path paradigm.

WEB-PATH PARADIGM

Collections of hypertext documents can be categorized by the underlying topology of links and nodes [Parunak, 1989]. WWW databases are intrinsically directed cyclic graphs. This can be thought of as a web-like structure (hence the Web in WWW). Yet most WWW databases reside on file systems that are explicitly hierarchical, e.g. UNIX™, Macintosh, VAX, etc. As a result of this incongruence, problems can arises when one attempts to view such databases. WebViz tackles this problem by displaying the database as a directed graph[2], with nodes representing separate documents in the database and links representing the hyperlinks, or paths, between documents. When a user "travels" from a source document to a separate destination document via the hyperlink embedded in the source document, a path is said to have been taken[3]. This path corresponds to the user clicking on the anchor point and retrieving the anchor (See Case One in Figure 1). We refer to this scenario as internal referencing (point of origin coming from within the database).

Note that since WWW enables users to enter a database via

1. For the purposes of this paper, the terms accesses and document requests will be used interchangeably.

2. The screen capture presented does not display arrows at the end of links. The data structure WebViz uses, however contains directional information. Arrows are soon to be implemented.

3. This contrasts to hyperlinks which point to different location with in the same document. WebViz does not analyze such information since such events are not captured by Hypertext Transfer Protocol (HTTP) servers.

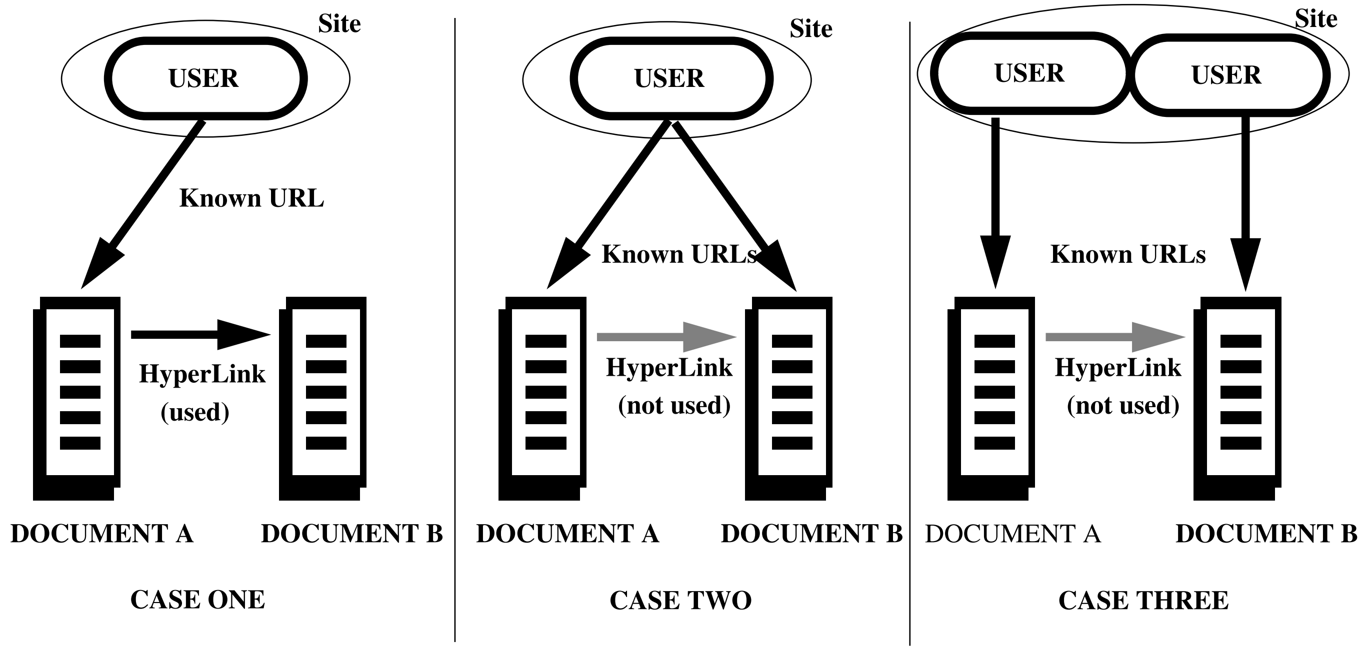

Figure 1: Three different access patterns

any document (via a known Uniform Resource Locator, or URL), causality between successive document requests is not always decidable. That is, even though there may exist a path between document A and document B and the access log records a request for document A followed by a request for document B from the same site, it remains a possibility that 1) the user at the site knew the location of both document A and document B and requested each file separately (See Case Two in Figure 1), or 2) there were two different users logged onto the site who happened to request document A and document B individually and in that order (See Case Three in Figure 1). That is the users did not click on the hyperlink in A to get to B. We refer to these scenarios as external referencing (point of origin exists outside the database) and dual referencing (points of origin in separate address spaces). Even though the possibility of other cases exists, WebViz assumes the Case One scenario for successive document requests. It is this assumption that underlies the algorithm for determining the paths taken by users in the access log.

WebViz uses the Web-Path paradigm to display the relations between the access log and the local database. Specifically, the program displays the documents of the local database and the connections between the documents as a web-like graph structure. Information is gathered from the access log about the number of times documents have been accessed as well as the recency of these accesses. WebViz further infers paths travelled by users by assuming that successive accesses by each user were internally referenced. The number of times paths were taken as well as the recency of the traversals are collected hence also by WebViz for display.

To recap, WebViz visualizes the collection of hypertext documents as a directed cyclic graph. The links in this web-like structure are referred to as paths, and represent the hyperlinks between documents. Nodes represent separate documents. Documents connected by hyperlinks can be successively accessed either internally or externally. By utilizing the Web-Path paradigm, WebViz collects frequency and recency information about documents and paths to drive the visualization. We now move onto an explanation of how WebViz creates the visualization. The following sections are arranged in the order that each stage is invoked during program execution.

INITIALIZATION

WebViz currently parses the National Center for Supercomputing Application's (NCSA) Hypertext Transfer Protocol (HTTP) 1.0 server access logs. As demonstrated by other access log analyzers, writing separate parsing routines for other HTTP servers is trivial. The sample access log entries below shows that the time of access, the machine name (either hostname or DSN), and requested file are logged for each transaction.

```
foo.gatech.edu [Tue Mar 8 10:50:25 1994] GET /gvu/intro_gvu.html HTTP/1.0
128.37.132.23 [Tue Mar 8 10:51:31 1994] GET /gvu/agenda.html HTTP/1.0
bar.gatech.edu [Tue Mar 8 10:52:01 1994] GET /gvu/agenda_more.html HTTP/1.0
```

Initially, lookups tables of hostname to DSN and DSN to hostname mappings are read into two separate hash tables. The intent here is to reduce the time consuming task of looking up a machine's DSN or hostname, since the log can contain either type of entry (see above example). Hence, the process of looking up hostnames and DSN numbers, which is network dependent and therefore potentially prohibitively slow, is done precisely once for each machine in the access log. Next, the specified access log is read into memory into a structure we refer to as the Master Log. With each transaction read, the hash tables are first consulted to see if the mapping is know and as a last resort, attempts the look up using the appropriate system calls. Once the entire access

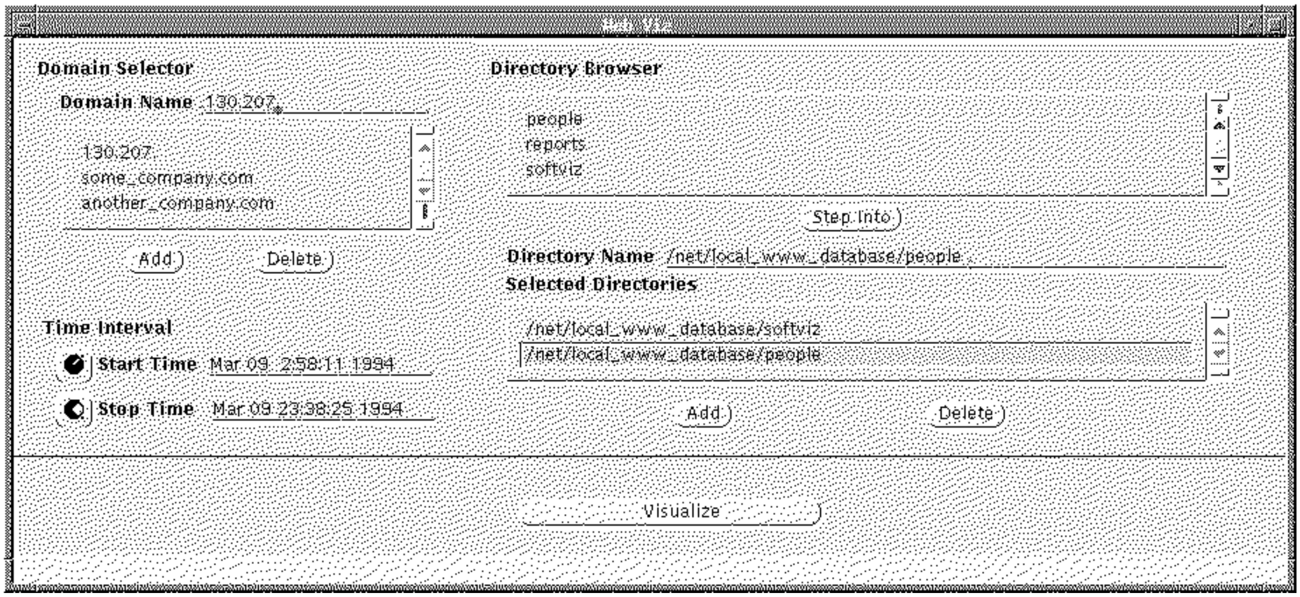

Figure 2: The View Control Window

log has been processed, the time of the first and last entry can be extracted from the Master Log for use in the View Control Window.

The View Control Window (see Figure 2) enables the user to determine the content of the visualization. Controls are provided the facilitate the selection of specific directories, domain names, and start and stop times. The directory selection allows for an arbitrary number of directories to be added to the visualization. As in the above example, lets assume that the user only wants to view the access patterns of the "softviz" and "people" directories, the person would add that directory to the selection list. This permits the user to restrict the contents of the web to only include the files within the specified directories, hence avoiding unnecessary files and directories. Because internally referenced documents are also added to the web, these files are also added. Thus, even though the user may requests to see only access patterns from a specific directory, additional files from other directories may be included into the visualization; however, embedded media (images, sounds, etc.) are not added.

Similarly, the domain selector enables the user to restrict the visualization to only machines that have accessed the database whose hostname or DSN contain the specified substring. This allows the end user to look at the access patterns from local machines, machines from specific companies, etc. (In the above example, we have restricted the view to three companies, two specified by hostname and the other by DSN). Clearly, unless complete or nearly complete DSNs are used, ambiguous results will occur, i.e. numerous machines will match and their accesses will end up in the visualization. Finally, the user can control the start and stop times used for the visualization. Hence, peak periods can be isolated just as easily as longer periods of time for analysis. To summarize, all the variable attributes recorded in the access log (time, machine making request, and requested file) are subject to user filtering.

Once the user has finished determining the view, the specifications are used to create a copy of the Master Log. This copy, called the View List, contains only the entries from the Master Log that the user desires to visualize. While this list provides enough information to determine the number of visits to a file and the times the file was accessed, it does not provide the times and the frequency path was travelled. This information is gathered by creating an Edge List that contains the source file, the destination file, the access times for both files, and the DSN of the machine traversing the path. A previously stated, an path is considered to have been travelled if there exists a path in the web, the same machine is making the successive requests, disregarding the possibility of external references (Case Two of Figure 1) and dual references (Case Three of Figure 1). We do place a time constraint on the interval between accesses of three days. That is, if the interval is greater than three days, we assume the user requested the document via another hyperlink that then one embedded in the source document. The selection of the three day period was not based on any empirical evidence. Once these lists have been created, the local database is processed.

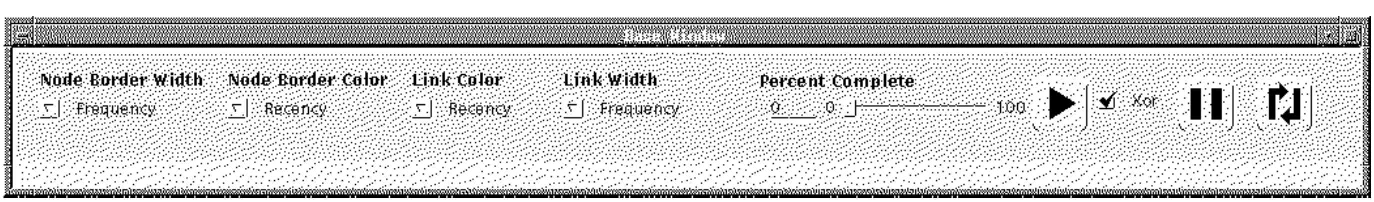

Figure 3: The WebViz Control Window

LOCAL DATABASE PROCESSING

The local database is processed to ascertain the structure of the web. The files in the database are processed one at a time, with processing proceeding recursively through the file system hierarchy. For each file being processed, if a corresponding node does not already exist in the web, a node is added. Currently, each node contains the file's name, size, and last modification time (additional information like owner, number and type of embedded media, etc. could be added to facilitate more sophisticated analyses). Files that do not contain Hypertext Markup Language (HTML) are not processed or added to the web at this stage. This decision reflects the implicit assigning of roles in HTML. That is, marked-up files act as either end documents or as intermediary documents with paths to other documents, while non HTML files can only assume end document roles. For each marked-up file, the contents are parsed and the URLs that point within the database are extracted (relatively addressed URLs are simplified into their full path names). If a node does not already exist for the file, a node is created and inserted into the web. Regardless of document type, a link is added from the processed document to the anchor, since it can be referenced internally and hence of possible analytical interest. At the end of the local database processing stage, the structure of the web has been defined as is ready to be displayed.

GRAPH LAYOUT

Graph layout is an arduous task in any setting - more so in WebViz since there are multiple, possibly conflicting interests:

Clarity The layout must good use of the available space to present the information in an easy to read fashion. Occlusion of nodes by other nodes or edges should be avoided.

Natural Structure Hierarchical graphs present a natural structure for embedding. The hierarchy in the web mirrors the file system hierarchy of the database as far as possible.

Presentation The graph must look presentable. Centering, regular spacing between nodes, staggering of nodes to avoid collinearity contribute to this end. A good presentation will try and minimize the lengths of edges in the graph. This may be incompatible with a hierarchical embedding since home directories which tend to be high up in the hierarchy have plenty of back edges and are best placed near the center of the graph.

A good layout will try and do justice to all these criteria in a judicious fashion. Since there is no clear optimization criterion many schemes are possible [Rivlin, 1994; Parunak,

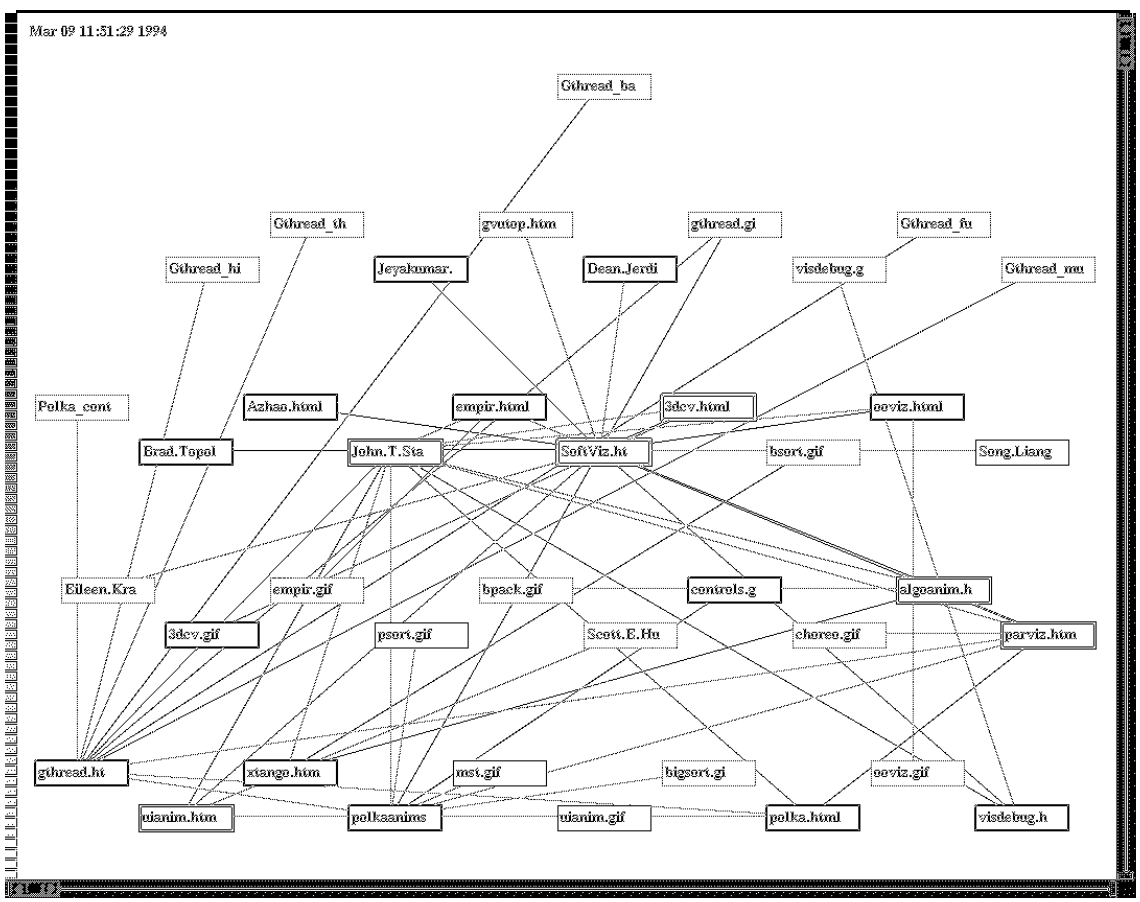

Figure 4: WebViz Screen Dump

1989]. The one we adopt presently is a randomized scheme with greedy placement of nodes. Besides being computationally cheap and easy to implement the randomization has an added benefit. If a certain embedding is not found satisfactory the scheme can generate a new graph for the user's consideration.

Specifically, our algorithm is as follows:

1. For each node we compute its "depth" in the UNIX™ file system hierarchy and use it to sort the nodes. Nodes are embedded in decreasing order of depth. As a result nodes that are high up in the hierarchy and have a lot of references will be placed close to their natural position.

2. The available screen space is partitioned into compartments of uniform size. The number of compartments is of the same order as the number of nodes in the graph. Each compartment will hold at most one of the nodes to be embedded. Note that the partitioning problem is a more complicated in 2D than it is in 1D.

3. Compartments are staggered at regular intervals in the X and Y direction to prevent collinearity.

4. For each node, twenty random empty compartments are sampled. The node is embedded in the compartment which minimizes a penalty function. The penalty function weights the following criteria:

   a) The Euclidean distance from the vertical line that partitions the screen space into two halves. Note that his criterion tries to keep the nodes close to the center of the screen.

   b) The Euclidean distance from a horizontal line that represents the natural position of nodes for the given depth value. This helps place nodes close to their natural position in the file system hierarchy.

   c) The Euclidean distance from all adjacent nodes that have already been embedded. This minimizes the length of edges, i.e. this function attempts to clusters associated nodes.

5. When the graph is drawn, edges (presently straight) are drawn before nodes to prevent occlusion. Occlusion of nodes by other nodes is avoided by the compartment scheme which also ensures moderately good usage of the available space.

The embedding produced by this scheme seems balanced and presentable. However, there is always room for other schemes - e.g. a tiered scheme that strictly follows the file system hierarchy or a scheme that minimizes edge intersections (see Future Work section below). We have used straight edges rather than curved edges to simplify "picking" and speed-up redraw, since the graph needs to redrawn for animation, as edges change thickness and color with the passage of time.

VISUAL MAPPING

Eventually in a visualization, data (processed or raw) needs to be mapped to visual (or audio) parameters. In our case the visual parameters are the thickness and color of nodes and links. We render labels in a fixed color to maintain readability. Thickness has a low resolution (4 levels currently) while color provides a much level of detail. The two parameters in each case are mapped to the either recency or frequency of access. Formally:

1. The recency of access of a node (link) is the time elapsed since the last access (traversal) of the node (link).

2. The frequency of access of a node (link) is the number of accesses (traversals) it has suffered since the beginning of the simulation expressed as a percentage of the maximum number of accesses (traversals) of any node (link) in the graph.

Since frequency and recency ranges tend to be large and it is desirable that the sensitivity of the mapping be greater for small values than larger values, we use a quasi-logarithmic function to map from four data ranges called "Quartiles" to the visual parameter range.

Recency	Quartile	Frequency
0 - 60 secs	First	100-51%
1 - 60 mins	Second	50-21%
1 -24 hrs	Third	20-6%
> 1 day	Fourth	6-0%

Table 1: Quartile Mappings for Recency and Frequency

To simplify computation we use a piecewise-linear curve, consisting of four linear segments. In the case of recency we map the real-time duration since the last access to the visual parameter. In case of frequency it is the number of accesses expressed as percentage of the maximum number of accesses in the log.

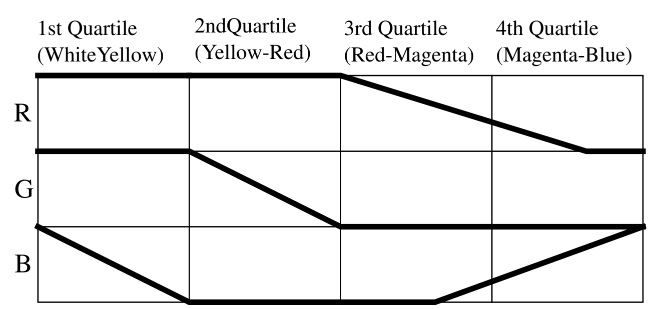

Figure 5: Color Map Organization

The 4 "quartiles" are mapped to colors as shown in the above figure. This intuitively mimics the non-linear cooling curve of a hot body; from white hot to yellow hot through red hot to blue. The initial variation is rapid and corresponds to the first quartile. The variation slows down gradually and never quite reaches the end of the 4th quartile. For the 4th Quartile we need a finite and reasonable upper bound to get some variation. Values larger than the upper bound map to the end of the 4th Quartile.

In the case of thickness, the thickness values 4, 3, 2 and 1 correspond to the quartiles 1, 2, 3 and 4 respectively.

Given the mappings in Table 1, we can understand the relationship between the visual attributes of nodes and links and their access history. A white-hot body is intuitively one that has just been touched (if color is mapped to recency) or very frequently accessed (if color is mapped to frequency). A blue-body was touched a long time ago (if color is mapped to recency) or very infrequently (if color is mapped to frequency). In the case of recency there is a real-world correspondence. A white-hot body must have been touched within the last minute. A blue body on the other hand is one that has not been touched for at least a day.

Thickness and color are typically mapped to separate quantities. If they are mapped to the same quantity we get redundancy, which can be beneficial too. Redundancy can help reinforce the expressiveness of the graph.

Now that the steps leading to and the mechanisms behind the visualization have been explained, we next discuss the actual visualization.

THE VISUALIZATION

The visualization is composed of two separate windows, the WebViz Control Window (see Figure 3) and the actual display window (see Figure 4[4]). The former provides the user with controls to adjust the bindings (the left-most buttons), select a specific time to view (the Percent Complete slider), control the animation (the play and pause icons), and rearrange the layout (the right-most button). The node and link binding buttons can be bound to either frequency or recency information. In Figure 3, the configuration is such that a node's width corresponds to how often the document was accessed and the node's color corresponds to the recency of the node's last access. Similarly, a link's width represents recency and the link's width represents frequency. The user can adjust these bindings as well as change the layout at time. Given an embedding on the screen the user can select a node or a link to get information about it.

Temporal manipulation is achieved by either the slider or by the playback controls. The slider enables the user to select a time to be displayed. the playback controls starting and pausing the playback of the events in the access log (recall that only the events selected by the user's filtering get displayed). Currently the playback is not synchronized with real-time. The simulation takes discrete steps forward in simulation time, and renders views of the graph in sequence. We are considering making the playback adaptive so that it completes in a user-specified interval of time.

4. WebViz displays are color. Unfortunately, the conversion process to incorporate the images into this document degraded the quality of the images, i.e. the screen capture of the display window looks dramatically different in paper than on screen.

FUTURE WORK

While WebViz achieves the goal of being a tool for WWW access log visualization, extensions to the interface, layout, and available analyses will make WebViz more robust. With respect to the View Control Window interface, we intend to create a timeline widget with two sliders and a rescale option to enable better time manipulation. Also, the directory and domain selectors could use some refinement. The WebViz Control Window needs controls for varying the speed of the simulation well adjusting the amount of time used to advance each frame. Clearly, since the current layout scheme does not allow for different layouts to be reproduced, mechanisms that enable the user to add and deleted preferred layouts will be developed. Along these lines, we are currently experimenting with additional layout methods. These include 1) a hierarchical layout that mirrors the organization of the database for a file system perspective, 2) a derivative of the multi dimensional scaling node placement algorithm [Eick, 1993], 3) a layout that uses a node's relative out centrality (ROC) as the primary placement determinant [Rivlin, 1994], and others. Direct manipulation of a node's placement will be added for all selected layout algorithms.

We are also experimenting with more sophisticated analyses. For instance, while it is useful to know how many times documents have been accesses and when, presenting the access information based upon "what's hot and what's not", (i.e. collapsing the recency and frequency information for each document and path into a quantitizable number), might also prove useful. This type of information answers the question: "What is the most popular document at a given moment?" To implement this, we are considering adding a connectionist form of short term memory, most likely a gamma based memory [Mozer, 1993].

Another tools under consideration is based upon initial evidence that the most frequently and recently accessed documents in a given time frame (i.e. day 0 through day 7) will be accessed on the day immediately following the time frame (i.e. day 8) [Recker, 1994; Anderson, 1991]. Hence, once the access pattern has been established for a database, subsequent access could be compared to the expected value and displayed as positive/negative deviations.

Finally, given that all salient information is readily available, WebViz can provided tab delineated file dumps for use by spreadsheets and graphing software. Hence, users can get output for specific files, paths, time intervals, etc. in a point and click manner. While other access log analyzers provide similar functionality as far as document accesses, WebViz provides path analysis in an interactive environment. Thus, users can visualize the data and isolate specific patterns before deciding upon dumping to file.

CONCLUSIONS

Motivated by providing useful analyses of WWW access logs, we developed WebViz. Towards this end, we enable database designers and maintainers to visualize the database's document space and re-issue events from the access log. By providing the user with controls to adjust the bindings properties of nodes and links, access patterns can be inferred. These accesses patterns can contribute to structural and contextual changes in the database.

ACKNOWLEDGEMENTS

The authors extend their appreciation to the Graphics, Visualization, and Usability Center and its members for their support and assistance, especially John Stasko.

REFERENCES

Anderson, John R. & Schooler, Lael J. (1991) Reflection of the environment in memory. *American Psychological Society, 2, 62.* 396-408.

Eick, Stephen G. & Willis, Graham J. (1993) Navigating large networks with hierarchies. *Proceedings of IEEE Visualization Conference, 1993.* 204-210.

Nielson, Jakob. (1990) The art of navigating through hypertext. *Communications of the ACM 33, 3.* 296-310.

Mozer, Michael C. (1993) In A. S. Weigend & N. A. Gershenfeld (Eds.). *SFI Studies in the Sciences of Complexity, Proc. Vol XV.*??? Addison-Wesley.

Parunak, H. Van Dyke. (1989) Hypermedia topologies and user navigation. *Hypertext '89 Proceedings.* 43-50.

Recker, Margaret M. & PItkow, James E. (in preparation) Predicting document access in large, multimedia repositories: a www case study.

Rivlin, Ehud & Botafogo, Rodrigo & Shneiderman, Ben. (1994). Navigating in hyperspace: designing a structure-based toolbox. *Communications of the ACM 37, 2.* 87-96.