WebMelody: Sonification of Web servers

Maria Barra Tania Cillo Antonio De Santis
Umberto Ferraro Petrillo Alberto Negro Vittorio Scarano

Dipartimento di Informatica ed Applicazioni "R.M. Capocelli", Università di Salerno
Baronissi (Salerno), 84081, Italy

1 Introduction

World Wide Web servers are growing in size, complexity and workload. It is often the case, for popular servers, that millions of requests are served per day. These requests are logged to files and, later, analyzed by the Webmaster so that they can provide useful hints on tuning and/or malfunctioning in the Web server. Given the workload, these log files are huge and the analysis is greatly helped (or in some cases made possible) by software programs that perform statistics on such massive amount of data, grouped by several criteria (time, location, Internet domain name, etc.).

Web servers, usually, offer the capability to choose the kind of information that is logged to files. It is a common situation, therefore, that monitoring a Web server is an off-line activity: performed on a regular basis, or prompted by evident malfunctioning.

We argue that it is important to provide an effective, efficient way to monitor in *real-time* the behavior of the server, especially for large Web servers that are used as a company asset. In fact, when the Web server is used not only as a presentation of the company but also to provide products information and, for example, to support users and clients, monitoring its behavior and access patterns is critical and should be performed as often as possible. In this way, prompt action can be taken by the Webmaster to solve malfunctioning or to fine-tune the system.

By real-time monitoring we mean that the system must provide feedback of an event to the Webmaster within a short, fixed time interval δ. Monitoring a Web server in real-time cannot be effectively achieved by usual log analyzers even if they claim "real-time" capabilities. In fact, they rely (mostly) on information provided by the log files that is assembled and presented in a variety of ways. Using log-analyzers "often" (hourly or more often) still has the drawback that information is presented to the Webmaster only if explicitly requested (client pull) and is not pushed to Webmaster's workstation. A possible exception is the capability to send alarm messages via messages on cellular phones (via SMS) or faxes but they are used only to inform of evident and severe malfunctioning (server and/or connection down).

Another pitfall of "real-time" log analyzers is that, being dependent on log files, requests that fail abruptly and do not generate log lines are not reported and heavy processing requests are reported only when the request is satisfied. We can say that log analyzers are able to monitor server throughput while a real time monitoring system must also provide information about server load.

As a consequence of being a real-time monitoring system, it is crucial that user interface is not invasive and highly personalizable. Our choice is to use the technique known as sonification[4, 6], i.e. the usage of nonspeech audio to convey information, since it is particularly useful when there is an abundance of data to be considered . Sonification of data is not new: the "ticking" of a Geiger counter as well as sonar is well known since decades.

Sonification is used successfully in several contexts and is, certainly, not a new technique: the "ticking" of a Geiger counter as well as sonars is well known since decades. Early application of sonification to computers are reported, by Giorgio Ausiello [1] about a prototype computer Olivetti 9104 at the IAC (*Istituto per le Applicazioni del Calcolo* of the Italian Council of Research (CNR)) with a garbage collector that used sounds with different tones to indicate the memory regions where the garbage collector was working. Also, Mike Muuss [5] used sonification in order to troubleshoot intermittent network connections by interfacing the output of a `ping` program to an audio device ("vocoder"): each time there was a failing link in the local network he administered, he moved in the building checking each segment of the network and listening to the audio output in his room and, when it stopped, it meant that the source of the intermittent behavior was found.

In general, sonification is particularly as "peripheral information" provider [3] being based on the consideration that screen space is, nowadays, a very limited resource while many are the parameters that the Webmaster may want to control about the Web server.

2 WebMelody: monitoring by sonification

We briefly describe here, the architecture of WebMelody, noticing also that the architecture is the base for several kind of clients (that are actually under development).

We envision several real-life situations where a real-time, and still non-completely absorbing, tool can help the Webmaster. Our system is completely and easily configurable and, for example, the Webmaster is able to early detect Denial of Service attacks, to determine that the server does not work, to recognize load peaks and have some feeling about possible explanations, to discern among the different types of requests (determining where the requests come from, or discriminating accesses by different agents, etc.), to monitor accesses to restricted (by user authentication) directories; to recognize the different error types and the HTTP method of the request.

The architecture of our prototype consists of three components: the sonification Apache module, the *Collector* server and the *WebPlayer* application.

1. The **sonification module** `mod_musical_log` is in charge of intercepting each HTTP request/response received by the Web server by, first, sending to the Collector the information about the arrival of a request. When the request is served, it also sends relevant information to the Collector according to its filtering of request/response HTTP parameters. The module is developed in C with the Apache API and is hooked into two phases of the processing request loop of the Apache server (called `post_read_request` and `logging`) in order to intercept the whole set of information that are available at the beginning and right at the end of the processing. The information are processed according to a set of sonification patterns grouped in "*Channels*". A Channel is a set of rules and directives describing which information are relevant for monitoring and how they must be translated into sounds.

2. The **Collector** is a Java application that works as middle level component of our architecture: it buffers the events provided by the Web server, parses them and then instruct the remote WebPlayer about the sounds to be played. It is in the position to monitor load and throughput of the Web server as well as make the analysis of the events that must be played only if a threshold is passed. Then, a command string specifying the sounds to be played is produced and sent to the WebPlayer.

The Collector is able to evaluate statistics on the workload and analysis of events with threshold, based on the output of all the children processes of the Apache server. But, by introducing a middle level we make easy the management of multiple players as well as make possible to use the Collector to receive data from other sources (servers of any type) and have the player work on different Channels coming from different servers.

Since the Collector has the task to collect and process the events according to the eventually specified thresholds, it works by examining batch of events at the interval specified in the configuration file (`StatisticsFrequency`). This means that an event happened on the server at time t is sent at most `StatisticsFrequency` seconds later to the WebPlayer.

3. The **WebPlayer** is a Java application and is the component of our architecture that produces the audio output. At startup, the WebPlayer downloads several sets of sound files from the network using the information provided by the `mod_musical_log` through the Collector. The loaded sound files are then played according to the command strings periodically received from the Collector.

The simultaneous playback of several sounds is implemented by a multiple audio-channel sound architecture. When WebPlayer starts, a set of audio tracks is loaded from a remote Web server, then each track reserves for itself a unique audio channel. While running, the WebPlayer determines what audio-channels are to be played according to the information received from the Collector.

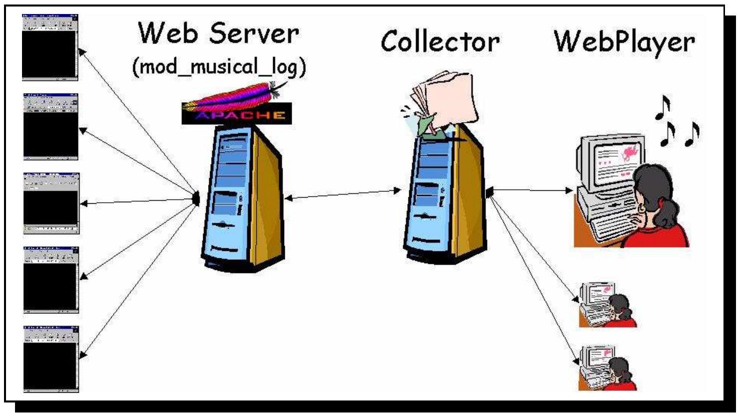

Figure 1: The architecture of WebMelody: the three levels.

3 Conclusions

Our system allows real-time monitoring of Web servers. Meaningful events (according to the configuration chosen by the Webmaster) are reported within at most `StatisticsFrequency` seconds to the Collector and then played $2 *$ `StatisticsFrequency` seconds later by the WebPlayer. This mechanism allows also monitoring the load of the server rather than only the throughput.

The system is highly configurable and versatile; efficiency is preserved by moving the most part of the computation away from the server. The interface (sonification) offers an unobtrusive representation of server behaviour to the Webmaster workstation.

Details on the project, as well as the WebMelody package are available at `http://isis.dia.unisa.it/SONIFICATION` where the interested reader can also download some test MIDI sound tracks, included the one specifically designed according to the musical considerations in this paper.

The work is actually under going along with the following guidelines. First of all, other plug-in clients such as e-mail, SMS and ordinary graph indicators are under development and are going to be included in the system architecture. Then, some changes in the Collector are planned in order to take information about multiple servers and convey it to the clients. Also, it is planned to adapt the architecture so that different servers (such as FTP and NFS servers) can be monitored in the same way.

Acknowledgments. The authors gratefully acknowledge Ugo Vaccaro, Alberto Marchetti Spaccamela and Giorgio Ausiello for information about the musical garbage collector.

References

[1] Giorgio Ausiello, Private communication as reported by Alberto Marchetti Spaccamela.

[2] M. Barra, T. Cillo, A. De Santis, U. Ferraro Petrillo, A. Negro, V .Scarano. “WebMelody: Sonification of Web Servers”. Poster Proc. of 9th International World Wide Web Conference (WWW9), Amsterdam (Holland) May 15-19 2000. Foretec Seminars.

[3] P. Maglio, C.S. Campbell. “Tradeoffs in Displaying Peripheral Information”. Proc. of ACM CHI 2000.

[4] Madhyastha, Tara M. and Reed, Daniel A. “Data Sonification: Do You See What I Hear”. IEEE Software, vol. 12 n. 2 pp. 46-56. 1995.

[5] Mike Muuss Homepage at `http://ftp.arl.mil/ mike/ping.html`

[6] “Sonification Report: Status of the Field and Research Agenda” Editorial Committee: Gregory Kramer, Chair; Bruce Walker, Project Coordinator; Terri Bonebright; Perry Cook; John Flowers; Nadine Miner; John Neuhoff.
`http://www.icad.org/websiteV2.0/References/nsf.html`