Using Transactional Web Archives To Handle Server Errors

Zhiwu Xie[1], Prashant Chandrasekar[2], and Edward A. Fox[2]
[1]University Libraries and [2]Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, VA
{zhiwuxie, peecee, fox@vt.edu }

ABSTRACT
We describe a web archiving application that handles server errors using the most recently archived representation of the requested web resource. The application is developed as an Apache module. It leverages the transactional web archiving tool SiteStory, which archives all previously accessed representations of web resources originating from a website. This application helps to improve the website's quality of service by temporarily masking server errors from the end user and gaining precious time for the system administrator to debug and recover from server failures. By providing pertinent support to website operations, we aim to reduce the resistance to transactional web archiving, which in turn may lead to a better coverage of web history.

Categories and Subject Descriptors
H.3.7 [**Information Storage and Retrieval**]: Digital Libraries

Keywords
Digital preservation; transactional web archiving; SiteStory; Memento.

1. INTRODUCTION
By estimates [1][2], existing web archives barely scratch the surface of the total web history. The low coverage may partially be attributed to the crawler-based archiving approach predominantly used by these archives. A web crawler can only archive the content it actively fetches through the scheduled crawling. However, the change of web resources is inherently unpredictable, making it extremely difficult to interleave the crawling schedule with the changes. Observing the politeness policy further limits the crawler's ability to track changes.

More comprehensive web history may be collected by involving more stakeholders through transactional web archiving [3]. A transaction is initiated by the user of a website. The archive sits between the user and the origin server and passively collects and archives the responses used to fulfill the user requests. The collective response to all these requests is a close approximation to a website's full history, or at least its memorable portion.

JCDL'15, June 21-25, 2015, Knoxville, TN, USA
ACM 978-1-4503-3594-2/15/06.
http://dx.doi.org/10.1145/2756406.2756955

Despite its advantages, archiving web transactions requires cooperation from the website owner. Only the origin server has information about all requests and responses; therefore the archive needs to be part of it. But it is not easy to engage website operations staff and convince them of the archive's value. Typical IT operations are preoccupied with their immediate needs and pay little attention to services whose benefits are longer term. It is therefore crucial for us to expand the value proposition of web archiving beyond the pledged altruistic cause and make transactional web archives immediately useful to day-to-day IT operations.

In this paper we present a web archiving application intended to improve website uptime, a core quality of service indicator for web operations. It takes the most recently archived representation of a web resource to handle application server failures. Webmasters benefit from this application because they gain precious time to recover from application server failures without disrupting the majority of their users' web experience. Archivists also benefit from it because the fine archival granularity resulting from transactional archiving is impossible to attain otherwise.

2. ARCHITECTURE
Figure 1 illustrates the architecture of this archiving application. It assumes the typical 3-tier web application made up of 1) a front-end server, assumed to be Apache, 2) an application server, and 3) an optional database server.

The system includes SiteStory [4], a transactional web archiving tool developed by Los Alamos National Laboratory. SiteStory has two components: mod_sitestory, an Apache module installed and configured as part of the Apache frontend server, and SiteStory Web Archive, a Java application run in a Tomcat container that uses Berkeley DB to store the archived web content.

The application developed in this project, mod_uws, is similar to mod_sitestory in that it is also an Apache module. It handles web disruptions that generate HTTP 5xx error codes. These errors usually occur behind the frontend, and result from application server failure, internal network congestion and disruption, and database server bottlenecks and failures. These problems are not uncommon, and are of great concern to webmasters. When these errors occur, we assume the frontend server is still alive and working properly to generate HTTP 5xx codes. This assumption is realistic because the commonly used gateway servers, e.g., Apache, are designed to handle high workloads and have well-designed scaling capabilities. They have been battle-hardened, and usually are more mature and robust than the other components in a web deployment.

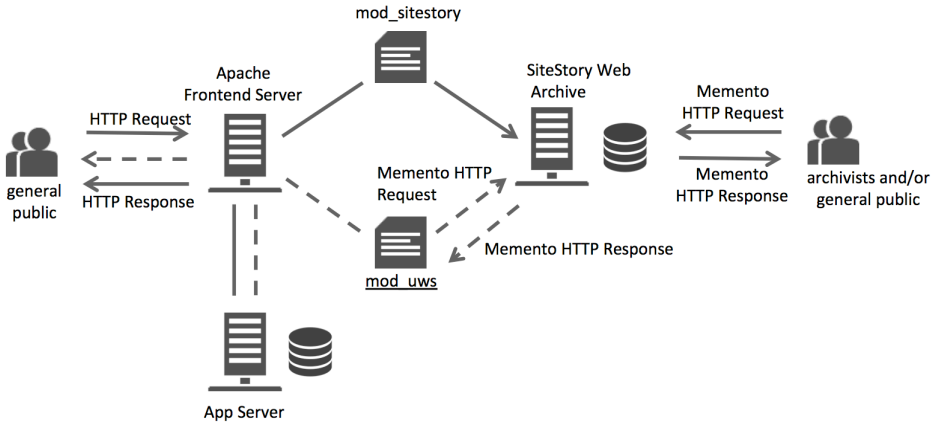

Figure 1. Architecture

In Figure 1 the solid line denotes the transactional archiving workflow under normal working conditions, while the dotted line presents the error-handling mode when an error occurs and an HTTP 5xx response triggers our application.

As explained in [4], under normal working conditions all HTTP 200 responses are sent in parallel to both the website user and the SiteStory web archive. This archive therefore always contains the most recent server state until an error occurs. At that point an HTTP 5xx response normally would have to be sent back to the requester through the Apache frontend server. However, instead, mod_uws will detect the error, become active, and intercept the 5xx response. Then mod_uws will send a Memento request [5] to the SiteStory archive to retrieve the most recently archived copy of the requested URI. This copy will be sent back to the requester with appropriate HTTP headers modified, hence masking the server error from end users. Although currently not implemented, in theory once mod_uws is activated, it may adaptively manage any subsequent requests using algorithms like exponential backoff. This would help flatten any potential peak load and allow the application server to recover from bottlenecks.

We developed mod_uws at the level of frontend server in order to make it agnostic to various programming languages, development tools, web frameworks, and database products used to build the website. This ensures the broadest possible adoption base. Any web operation using Apache HTTP server can easily integrate and benefit from this application without significantly modifying its deployment and configurations.

3. FUTURE WORK

It has been shown that using SiteStory does not significantly affect the performance of Apache HTTP server [6]. We will soon publish the performance test results showing how the current implementation of mod_uws affects Apache. We will also attempt to put the Memento request and response function into an external application in the hope of lessening the performance impact of mod_uws on Apache.

Larger web operations usually deploy a load balancer in front of many Apache servers. This provides us with the opportunity to move both SiteStory and the error handling application one level up to the load balancer to further improve the archiving efficiency and the ease of deployment.

4. ACKNOWLEDGMENTS

This work is supported in part by the Web Archiving Incentive Awards, funded as part of Columbia University Libraries' 2013 Mellon Grant for Collaborations in Web Content Archiving.

5. REFERENCES

[1] SalahEldeen, H.M. and Nelson, M.L. 2012. Losing My Revolution: How Many Resources Shared on Social Media Have Been Lost? *Theory and Practice of Digital Libraries*. P. Zaphiris, G. Buchanan, E. Rasmussen, and F. Loizides, eds. Springer Berlin Heidelberg. 125–137.

[2] Ainsworth, S.G., AlSum, A., SalahEldeen, H., Weigle, M.C. and Nelson, M.L. 2012. How Much of the Web Is Archived? *arXiv:1212.6177 [cs]*. (Dec. 2012).

[3] Masanès, J. ed. 2006. *Web Archiving*. Springer.

[4] SiteStory Web Archive: http://mementoweb.github.io/SiteStory/index.html. Accessed: 2015-02-01.

[5] Van de Sompel, H., Nelson, M. and Sanderson, R. 2013. *HTTP Framework for Time-Based Access to Resource States–Memento*. IETF RFC 7089.

[6] Brunelle, J.F., Nelson, M.L., Balakireva, L., Sanderson, R. and Van de Sompel, H. 2013. Evaluating the SiteStory Transactional Web Archive with the ApacheBench Tool. *Research and Advanced Technology for Digital Libraries*. T. Aalberg, C. Papatheodorou, M. Dobreva, G. Tsakonas, and C.J. Farrugia, eds. Springer Berlin Heidelberg. 204–215.