WPBench: A Benchmark for Evaluating the Client-side Performance of Web 2.0 Applications*

Kaimin Zhang
University of Science and Technology of China, Hefei, China
coming@mail.ustc.edu.cn

Lu Wang, Xiaolin Guo
Tsinghua University
Beijing, China
{coolwanglu,ringoguo}@gmail.com

Aimin Pan, Bin B. Zhu
Microsoft Research Asia
Beijing, China
{aiminp,binzhu}@microsoft.com

ABSTRACT
In this paper, a benchmark called WPBench is reported to evaluate the responsiveness of Web browsers for modern Web 2.0 applications. In WPBench, variations of servers and networks are removed and the benchmark result is the closest to what Web users would perceive. To achieve these, WPBench records users' interactions with typical Web 2.0 applications, and then replays Web navigations when benchmarking browsers. The replay mechanism can emulate the actual user interactions and the characteristics of the servers and the networks in a consistent way independent of browsers so that any browser compliant to the standards can be benchmarked fairly. In addition to describing the design and generation of WPBench, we also report the WPBench comparison results on the responsiveness performance for three popular Web browsers: Internet Explorer, Firefox and Chrome.

Categories and Subject Descriptors
D.2.8 [Metrics]: Performance measures

General Terms
Measurement, Performance

Keywords
Web, Benchmark, Browser, Replay, JavaScript

1. INTRODUCTION
In recent years, dynamic content, scripting and asynchronous technologies like AJAX (Asynchronous JavaScript and XML) are widely adopted, leading to much more powerful functionalities in Web applications such as storage capacity, complex computation, and rich interactions with users. This trend leads to rich client-side experiences which can compete with conventional client-side software. The popularity of Web 2.0 applications such as Google Maps and Live Search has shown the feasibility of such a kind of Web browser-based applications. Along with the popularity, browsers, as one type of application platform, have attracted more and more attentions. A new "browser war" has already started to compete on the performance for Web 2.0 applications, e.g. the speed of JavaScript engine [1][2].

To study the performance of Web browsers for Web 2.0 applications, a benchmark representing typical Web 2.0 applications is needed. There already exist several benchmarks used to measure the performance of certain modules of a browser, such as SunSpider [3] for evaluating JavaScript engine and GUIMark [4] for layout engine. There also exist some benchmarks to measure the overall performance of a browser by using full-fledged Web applications [5] or with a set of existing benchmarks [6]. We have not yet seen a benchmark to evaluate browser's performance for typical Web 2.0 applications.

In this paper, we report the benchmark called *WPBench* (Web Performance Benchmark) that we have recently designed and developed to measure the performance of browsers for Web 2.0 applications. Instead of artificially constructing Web content based on a model of typical Web 2.0 applications, WPBench uses the real data from users' actually browsing and interacting with Web 2.0 sites. Variations of user interactions and network transmissions have been removed. Therefore WPBench produces a fairer benchmark for different Web browsers.

In WPBench, user interactions are recorded when users are browsing a set of the most popular Web 2.0 applications. These interactions are emulated during benchmarking browsers by instrumented JavaScript which is independent of Web browsers. Since the Web content, user interactions, and networking are exactly the same for these browsers, WPBench produces benchmark results fair to different Web browsers. Its responsiveness performance is closer to users' perception than any of other benchmarks. To the best of our knowledge, there exists no previous benchmark which can automatically emulate the process of user Web surfing in a way fair to Web browsers. We believe that a benchmark like WPBench is useful to evaluate the performance of Web browsers for modern Web 2.0 applications. By using WPBench, we also report the comparison results of the responsiveness performances of three popular Web browsers, Internet Explorer, Firefox, and Google Chrome.

The main contributions of this paper are a responsiveness benchmark fair to different Web browsers and nearly the same as what users would perceive in actual browsing, a novel replay mechanism by using JavaScript code embedded in the recorded Web pages with which the user interactions could be emulated, and a comparison of popular Web browsers for the interactive performance which is the closest to users' actual perception.

2. BENCHMARK DESIGN
2.1 Benchmark Requirements
For a benchmark to be able to evaluate the performance of Web browsers that users perceive, we argue that the Web pages in the benchmark must come from typical real Web sites, or have the same or similar characteristics as real Web sites. Since our benchmark is targeting at Web 2.0 applications, the characteristics of the interactions between Web users and pages must be considered in the benchmark. The benchmark should meet the following requirements:

*This work was done when Kaimin Zhang, Lu Wang, and Xiaolin Guo were interns at Microsoft Research Asia.

WWW 2009, April 20-24 2009, Madrid, Spain.
ACM 978-1-60588-487-4/09/04.

- The Web pages used in benchmarking the performance must be representative for Web 2.0 applications.
- The Web navigation sessions in the benchmark must be deterministically replayed exactly the same by any Web browser compliant to the standards.
- Variations of Web servers and networks should be avoided in the benchmark.

We build our benchmark by using the Web content from typical real Web 2.0 sites. The actual Web content is recorded by a proxy, and then standardized and instrumented for replaying during benchmarking browsers. Users' interactions with Web browsers and navigations between Web pages are emulated by JavaScript, which is independent of any Web browsers. Our benchmark meets all the aforementioned requirements.

2.2 WPBench Architecture

Figure 1 shows the architecture of WPBench, which consists of recording and replaying stages. In the recording stage, all the Web data as well as the events triggered by user's actions, including event types and timestamps, are recorded while users are surfing the Internet. Then the original Web data, including HTML, images, CSS (Cascading Style Sheets), etc., are processed to ensure compliance with the standards. The Web pages are instrumented by adding JavaScript code to emulate the recorded events and navigations accurately during replaying. The resulting data are saved in WPBench Store to be used in the replaying stage.

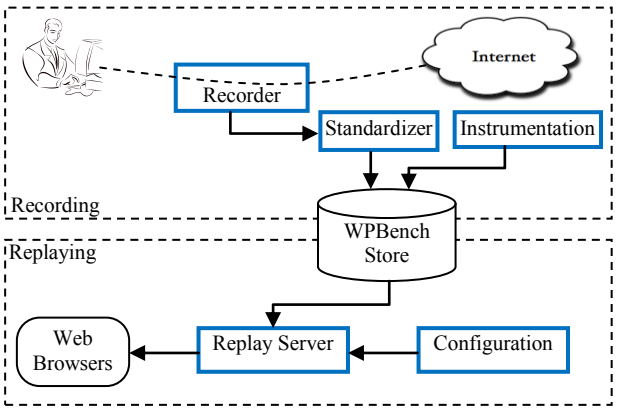

Figure 1. The Architecture of WPBench.

In the replaying stage, the data in WPBench Store are fed to browsers by a proxy according to the local configuration so that browsers could obtain the Web content as if they were actually from the Internet. The configuration can determine the replay policies, such as whether to emulate the networking latencies. During replaying, the user events and navigations are automatically emulated by JavaScript, resulting in the same behaviors for any browser compliant to the standards. Furthermore, the Web content is fetched from a proxy. Therefore, variations of Web servers and networks are effectively removed.

3. WPBench Generation

We selected 10 Web sites from the top 100 Web 2.0 applications for 2008 listed in *www.webware.com*, which were voted by millions of Internet users. They are *google.com*, *amazon.com*, *flickr.com*, *maps.live.com*, *yahoo.com*, *wikipedia.org*, *youtube.com*, *live.com*, *last.fm*, and *deviantart.com*. Each selected Web site represents a category of Web 2.0 applications or a typical usage in real life.

Then we organized a group of users in our lab to browse these Web sites in a way appropriate to the sites, e.g. listening and searching music for an audio site and finding locations for a map site. One navigation session was recorded for each selected site. Before recording, the local cache was cleared and the connection was set to direct to our proxy so that the proxy could record all the accessed Web data and insert JavaScript into the Web pages for the browser, which was Firefox in our experiments, to record the user interactions on the client side. The system was connected to the Internet through the Microsoft enterprise network.

The recorded data were merged and processed to remove the elements and components that don't follow the standards. JavaScript code is then added for emulating the recorded user interactions accurately during replaying. The resulting data were used to generate benchmark results. The data was 38MB, including 69 unique Web pages, 2687 files, and 1092 events. The total browsing time was 33 minutes.

The overhead of both recording and replaying were measured with a Firefox add-on (Firebug). 6.8% extra time relative to the total JavaScript code execution time was incurred in the recording stage, and 6.2% in the replaying stage.

4. Comparison of Browser Performance

We have used WPBench to compare the performance of several popular browsers in the market: Internet Explorer 7 (IE7), Internet Explorer 8 (IE 8 beta 2), Firefox 3.0, Firefox 3.1 (beta), and Google Chrome (beta). These browsers cover the most well-known layout engines, such as Trident and Gecko, as well as several widely used JavaScript engines. All these browsers can browse all the Web sites in WPBench normally except that IE 8 beta and Firefox 3.1 beta cannot browse one of them due to unsupported features used by the Web site. Formal releases of these two broswers are expected to fix these problems.

We employed two responsiveness metrics in the comparison: Page Loading Time and Event Responding Time. For both responsiveness metrics, Firefox 3.1 (beta) is the fastest. The order of the remaining browsers is: Firefox 3.0, Google Chrome, IE 8 (beta), and IE 7, in the order from the fastest to the slowest. The fastest browser, Firefox 3.1, is about 30% faster than the slowest one, IE 7.

5. REFERENCES

[1] JavaScript:TraceMonkey. https://wiki.mozilla.org/JavaScript:TraceMonkey

[2] V8 JavaScript Engine. http://code.google.com/p/v8/

[3] Sunspider JavaScript Benchmark. http://www2.webkit.org/perf/sunspider-0.9/sunspider.html

[4] GUIMark. http://www.craftymind.com/guimark/

[5] C. Amza, A. Chanda, A. L. Cox, S. Elnikety, R. Gil, K. Rajamani, W. Zwaenepoel, E. Cecchet, and J. Marguerite. Specification and Implementation of Dynamic Web Site Benchmarks. In 5th IEEE Workshop on Workload Characterization, 2002.

[6] J. Nielson, C. Williamson, and M. Arlitt. Benchmarking Modern Web Browsers. In 2nd IEEE Workshop on Hot Topics in Web Systems and Technologies, 2008.