Preliminary Measurements on the Effect of Server Adaptation for Web Content Delivery

Balachander Krishnamurthy, Craig Wills, Yin Zhang

Abstract—**The phrase "time-to-glass" has been used to describe the delay between the user's browser click and the rendering of the response from a Web server on the user's screen. A Web site has a strong incentive to reduce this time-to-glass to retain users who may otherwise lose interest and leave the site. Our work reports on a measurement study carried out from diverse client sites around the world with various levels of connectivity to the Internet. The study measured the impact of a variety of Web server actions for reducing the time-to-glass for retrieving content from a canonical Web site under our control. Our work examines various aspects affecting a client (such as low bandwidth or high latency) in the context of a suite of server actions, ranging from selecting a lower quality version of the resource to altering the manner of content delivery or varying server policy to keep connections open longer. We examined a large proxy log and extracted a subset of popular pages with diverse characteristics to construct our canonical site. By studying numerous performance related factors in a single unified framework and examining both individual actions as well as combination of actions, this first such study shows the efficacy of various server actions. Our results show that some actions have less impact than previously thought while others have significant impact for a class of clients with particular connectivity characteristics.**

I. Introduction

In earlier work we presented a way to characterize Web client's connectivity information [3]. A Web client may experience poor performance due to low bandwidth, high latency, network congestion, delay at intermediaries between the client and server, etc. A server can take remedial actions such as selecting a lower quality version of the resource for delivery or by altering the manner of content delivery. In this followup work, we present early measurement results on the actual latency reduction for a wide class of real and geographically dispersed set of clients.

Earlier research work has examined Web performance from the viewpoint of individual improvements in reducing user-perceived latency or load on the network. The set of ideas includes compression and delta encoding [5], the use of content distribution networks [1], examining impact of various protocol variations of HTTP [2] and bundling resources [8]. What these works have in common is the use of a single idea to explore impact on Web performance. Each of these pieces of research differ in their evaluation environment in the sense that they use different methodologies, workloads, and validation techniques.

There are three key differences in this work:

1. We examine multiple performance related factors in a single unified framework.
2. We examine a set of single actions that a server can take to improve performance as well as a meaningful combination of actions.
3. We use a canonical set of container documents with various distributions of embedded objects in terms of number and size. This approach allows the results of our work to be applied by a wide variety of sites to test the potential improvement of clients that visit them.

We explore the impact of various potential performance improvements by an active measurement testbed consisting of clients with different connectivity sending requests to a small number of Web servers under our control. Having control over the Web server and content allows us to examine the different performance components in an automated fashion. By downloading the canonical container document set via clients with different connectivity capabilities, we can measure the actual improvement as a result of various server actions. The following summarizes the server actions we investigate in this work for improving the response time for clients.

- *Altering the Content*. Given a range of content variants, a server could choose a reduced version for poorer clients, by including fewer, if any, embedded objects or by including "thinner" variants of embedded images.
- *Using a Content Distribution Network (CDN)*. In cases where the round-trip latency between client and Web site server is high, the use of CDNs by the Web site may yield lower response time for the client [1].
- *Altering Manner of Delivery of Content*. The content can be compressed with a suitable compression algorithm. The server can also bundle the embedded objects into a single resource, which can be retrieved by clients to avoid multiple rounds of requests to fetch the objects [8]. This

Krishnamurthy and Zhang are with AT&T Labs–Research, Florham Park, NJ, USA. Wills is with Worcester Polytechnic Institute, Worcester, MA, USA.email:{bala,yzhang}@research.att.com, cew@cs.wpi.edu. Contact author: Balachander Krishnamurthy, Fax: 973-360-8077, 180 Park Avenue, Florham Park, NJ 07932.

technique can be combined with compression to reduce the size of the bundle.

• *Maintaining Persistent Connections*. In HTTP/1.1, connections between a client and a server can persist beyond a single request-response exchange. If a server knows that a client has poor connectivity, it might wish to keep the connection open longer than usual to ensure that the client does not pay the overhead of having to tear down and set up a new connection.

II. Methodology

We want to investigate actions that a Web server can take to reduce download time for a client. While dynamic generation of Web content can take a nontrivial amount of time, the generation is under control of the server and can be improved independent of content delivery. The frequency and amount that content changes can significantly affect the usefulness of caching and delta encoding, but these techniques are only relevant for repeat accesses by a client or cluster of clients for a page. For now, we focus on characterizing pages based on the amount of content of a page because these affect the first access by a client for the page. We plan to consider frequency and amount of change in future work.

To characterize the amount of content for a page we examine the number of bytes in the container object, the number of embedded objects and the total number of bytes for the embedded objects. These three characteristics are used to classify pages in our work. We are interested in investigating which server actions are the most effective in reducing download times for different combinations of characteristics that a page may have.

Using these characteristics to classify Web page contents, the first part of our work was to identify Web content that covered the "space" of these characteristics. We wanted to populate a canonical test site with realistic content that we know is requested by clients. This test site would be used by clients to retrieve content in the context of different server actions.

We used recent (December 2001) proxy logs from a large manufacturing company with well over a 100,000 users. We only examined requests to the container object of a page by looking for HTML URLs and selected the 1000 most popular pages. We used a Web page retrieval tool in April 2002 to download each container object and embedded objects (frames, layers, cascading style sheets, javascript code and images) to determine the size of these objects. Objects referenced as a result of executing embedded javascript code were not considered.

From the initial list, 641 URLs containing one or more embedded objects were successfully retrieved. We used the 33% and 67% percentile values to create a small, medium and large value range for each characteristic. These ranges are shown in Table I. Using these three ranges for each of the three characteristics defines a total of 27 "buckets" for the classification of an individual page.

TABLE I
Ranges for Each Page Characteristic

Characteristic	Small	Medium	Large
Container Bytes	0-12K	12K+-30K	30K+
# of Embedded Objects	1-6	7-22	23+
Embedded Bytes	0-20K	20K+-55K	55K+

Using these ranges we determined the percentage of pages that fell in each bucket. These are shown in Table II. The table shows that 20% of these pages have a small number of container bytes, a small number of embedded objects and a small number of embedded bytes. 7% of the pages fall in the medium range for each characteristic and 14% fall in the large range for each characteristic.

TABLE II
Percentage of Pages in Each Characteristic Bucket Based on Popular Pages from Proxy Log

	Embedded Bytes								
	Small			Medium			Large		
Embedded	Cont. Bytes			Cont. Bytes			Cont. Bytes		
Objects	S	M	L	S	M	L	S	M	L
Small	20	6	2	4	1	0	0	0	0
Medium	2	3	1	5	7	8	2	5	3
Large	0	0	0	1	2	4	1	8	14

As a comparison we also looked at the home pages of 131 popular [4] Web sites using the same bucket ranges as defined in Table I. The percentage of home pages with characteristics defined in Table I is shown in Table III. In general, these pages have a larger number of embedded objects and bytes than the popular pages from the company proxy log. In terms of the project, this distribution simply indicates that the effect of server actions for more embedded content is of greater interest for popular site home pages.

While the percentage of pages in each bucket is interesting we primarily defined the ranges so that we could identify pages that spanned the space of all possible characteristics. To do this we selected two representative pages from within each bucket of the proxy log pages shown in Table II. In buckets containing many pages we tried to se-

TABLE III
PERCENTAGE OF PAGES IN EACH CHARACTERISTIC BUCKET BASED ON HOME PAGES FROM POPULAR SITES

	Embedded Bytes								
	Small			Medium			Large		
Embedded	Cont. Bytes			Cont. Bytes			Cont. Bytes		
Objects	S	M	L	S	M	L	S	M	L
Small	7	1	0	3	1	0	0	0	0
Medium	1	0	3	7	11	5	3	3	4
Large	0	0	0	0	3	12	7	11	18

lect two pages that were representative of characteristics within the bucket. Given that some buckets contain only one (2 buckets) or no page (4 buckets) we were able to select 44 pages to cover the space of characteristics.

The contents of these 44 pages were downloaded to files at a test site using the Web utility *wget* [7]. The utility automatically localizes links to embedded and traversal links, although it does not identify objects such as style sheets and javascript objects. These had to be additionally downloaded and in cases where 302 Found (redirection) responses were encountered, URLs in the container object were changed to match the downloaded name.

Once the content for the 44 test pages was ready on the test site, additional objects were created in preparation for testing the various server actions. A compressed version of each container object was created using the *gzip* utility. All of the embedded objects for each page were bundled into a single bundled object. A separate compressed bundle object was created using *gzip*. To test the effect of offloading embedded content from a server to a CDN, we crawled the content of a Web site known to use a major CDN[1] and cataloged a large number of objects along with sizes served by this CDN. We then matched each object at our test site with a similar size CDN-served object. These CDN-served matched objects were used to emulate the downloading of embedded content from a CDN server rather than the test site server.

We installed the test site on two relatively unloaded servers: `www.cs.wpi.edu` located on the East Coast of the U.S. and `www.icir.org` located on the West Coast. For client testing, we used *httperf* [6], to make automated retrievals to the test server and CDN site for testing of the various server actions. While retrieval using a real browser might be more realistic in measuring "time-to-glass," the use of *httperf* allows us to automate and control the retrieval under various conditions.

[1] We did not use the AT&T CDN to avoid appearance of bias.

III. EXPERIMENTS

Just as we deliberately chose Web content for our test Web site to include a variety of characteristics, we located clients with different connectivity characteristics to the test sites. We tested from clients in six locations:

1. att: AT& Labs–Research, New Jersey, USA,
2. de: Saarbruecken University in Germany,
3. isdn: home user in New Jersey, USA connected via a 128Kbps ISDN line,
4. modem: home user in New Jersey, USA connected via a 56Kbps dialup modem,
5. uk: London, England via a dedicated 56Kbps line, and
6. au: University of Melbourne, Australia.

We used measured round-trip time (RTT) and throughput to characterize the network connectivity characteristics between these clients and our test sites. The round trip time was determined using the average TCP connection setup time for the first object retrieval of each *httperf* test. We computed the average throughput for retrieving each of the bundle objects in the test site. We used these objects because they contained a larger number of bytes. The round-trip times and throughput for each of our client/server pairs are shown in Figure 1. Note: the att.wpi pair has a RTT of 9ms with a throughput of 462.6KB/sec.

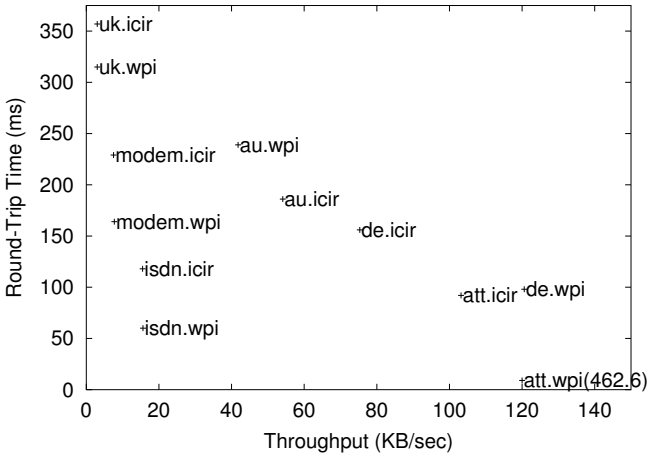

Fig. 1. Client/Server Connectivity

For a client, a set of tests for different server actions was run four times spaced over 24 hours with results shown being the average of these four time periods. Note: The uk client took so long to run the set of tests that only one set was run. We used the retrieval of a page from a test site using up to four parallel non-persistent HTTP/1.0 connections as a baseline measure because it requires no particular server actions. We refer to this approach as para-1.0. A real browser may begin retrieving embedded objects before retrieval of the container object is finished. However,

httperf does not begin retrieving the embedded objects until the container object has been retrieved. This separation does allow us to consider the effect of an action such as compression to reduce retrieval time of the container object independent of the effect of actions to reduce retrieval time for the embedded objects.

Using para-1.0 as a baseline we studied the relative effect of each of these server actions where the container object is retrieved first:
1. compress—retrieve container object in compressed form and embedded objects (uncompressed) using up to four parallel HTTP/1.0 connections
2. serial-1.1—serialized requests are made for embedded objects using up to two persistent HTTP/1.1 connections
3. pipe-1.1—pipelined requests made for embedded objects using a single persistent HTTP/1.1 connection
4. cdn—embedded objects retrieved from a CDN server using up to four parallel HTTP/1.0 connections
5. bundle—retrieve a single bundle of embedded objects.
6. baseonly—retrieve the container object, but no embedded objects. This action is intended to measure the potential response time savings for removing all embedded content.
7. halfobject—retrieve (top) half of the embedded objects using up to four parallel HTTP/1.0 connections; intended to examine the effect of removing some embedded objects from the container object.
8. halfembed—retrieve the top half of each embedded object; an action derived from the results of the para-1.0 retrieval that assumes it takes half the time to retrieve the data of each embedded object; intended to examine the effect of creating thinner versions of each embedded object.

Many of these actions can also be used in conjunction with each other. For example, we examined the impact of compressing the bundle object (bundle.gz) or using the CDN to serve the bundle object (cdnbundle). These two combinations are included in the results. In addition, the compression of the container object can be combined with any of the other actions for a cumulative effect. In our results, we note when such a combination leads to significant performance improvements.

Actions add to the costs of the server. The server has to create and store thinner variants of some objects; it must generate or pre-compute and store compressed or bundled content. These costs however can be amortized across multiple requests.

IV. Results

The results from our tests show a clear distinction between client/server pairs with relatively low throughput (uk, modem and isdn) and those with high throughput (att, au and de) to the two servers. There is less distinction based on the RTT between client and server. We use the throughput distinction in organizing the results.

A. Low Throughput Clients

Table IV shows the results for the isdn.wpi client/server pair. The table shows the results of actions for 8 of the 27 buckets of content mix shown in Tables II and III. These buckets represent 71% and 74% of the actual pages for the respective data sets. Space limitations prevent showing results for all 27 buckets. The top row in the table shows the average time in seconds to retrieve a page with the given mix of content. The remaining actions are divided into lossless, which do not change the content served to the client, and lossy, which change the content served.

The results show that the time to retrieve a page with large container page, a large number of embedded objects and a large number of embedded bytes is 10.49 seconds. Subsequent lines in the table show the relative percentage improvement if the various actions are taken. For emphasis on significant differences, only server actions that yield greater than 20% ('+') improvement for the majority of buckets are shown. Those yielding greater than 50% improvement are shown with '++'. Thus using a compressed version of the container object for this bucket saves 20-50% of the 10.49 seconds.

Overall the results show that compression of the container object has a significant effect for buckets with larger container objects, but the other lossless actions do not have significant effects and are not shown in the table. The combination of first bundling then compressing the embedded objects also leads to a significant improvement in download time. The action of serving only the container object yields a significant cost savings of greater than 50% for all content mixes. Serving only half of the embedded objects also has a significant effect, but trying to reduce the size of embedded objects while still serving the same number of objects has little positive effect.

Table V shows the results for the uk.icir pair, a low throughput, high latency client/server pair. Despite the difference in throughput and sharp contrast in latency, the relative impact of server actions is similar to isdn.wpi. Overall results for the other client/server pairs with low throughput shown in Figure 1 are similar. In particular, the use of a CDN for all of these client/server pairs did not yield at least 20% improvement because any latency reductions between a client and chosen CDN server relative to the test site server did not overcome the bandwidth constraints of the client.

TABLE IV
ISDN.WPI—ACTIONS WITH SIGNIFICANT IMPROVEMENTS OVER PARA-1.0 (+: 20-50%, ++: >50%)

	Container Bytes-Embedded Objects-Embedded Bytes							
Action	S-S-S	S-M-M	M-M-M	M-M-L	M-L-L	L-M-M	L-L-M	L-L-L
para-1.0	1.35s	3.50s	4.73s	7.13s	8.95s	6.27s	7.32s	10.49s
compress			+			+	+	+
bundle.gz		+	+	+	+	+	+	+
baseonly	++	++	++	++	++	++	++	++
halfobject	+	+	+	+	+		+	+

TABLE V
UK.ICIR—ACTIONS WITH SIGNIFICANT IMPROVEMENTS OVER PARA-1.0 (+: 20-50%, ++: >50%)

	Container Bytes-Embedded Objects-Embedded Bytes							
Action	S-S-S	S-M-M	M-M-M	M-M-L	M-L-L	L-M-M	L-L-M	L-L-L
para-1.0	6.76s	16.84s	22.96s	36.15s	42.94s	30.47s	33.01s	50.46s
compress			+			+	+	+
bundle.gz	+	+	+	+	+	+	+	+
baseonly	++	++	++	++	++	+	++	++
halfobject	+	+		+	+		+	+

B. High Throughput Clients

Table VI shows results for the att.icir client/server pair, which has high throughput. Unlike the low throughput clients, the relative improvement for compression is not significant, but the impact of actions such as the pipelining, use of a CDN and bundling have a significant effect, particularly for buckets with more content. For the att client, the RTT to the CDN server is approximately one-fourth of the value to the icir server with a throughput five times as much.

Table VII shows results for the au.wpi client/server pair with relatively high throughput but a much longer latency. Despite these differences, the tone of the results is the same as shown in Table VI. For this client, the relative RTT and throughput performance to the CDN server relative to the WPI server is an order of magnitude.

The other high throughput client/server pairs shown in Figure 1 yield similar results, although the impact for a CDN serving embedded objects is not a significant improvement for the att.wpi pair. This result is because the RTT from client to CDN server is actually larger than that between client and the WPI server.

V. CONCLUSIONS AND ONGOING WORK

This is the first study that we are aware to look at the impact of server actions for a variety of content and client conditions where each action is measured on a common platform. We are also able to evaluate the cumulative effect of two or more actions. Administrators of high volume websites can benefit from our results by examining their content mix to see how different actions will benefit their clients. The summary of the results of our work thus far is:

- Compression of HTML content is not universally useful. We did not find that compression had a significant effect on reducing response time for well-connected clients. Compression is an effective action when the client is bandwidth-constrained.
- The CDN was useful for improving performance of well-connected clients, but not so for bandwidth-constrained clients even when it provided a lower RTT for clients. This result extends what we found in [1]. Note that these comparisons were made with relatively unloaded servers.
- Persistent connections with serialized requests do not provide a significant performance improvement under a wide variety of client/content conditions.
- Persistent connections when combined with pipelining are only significant for high bandwidth clients.
- Bundling content, like pipelining, has some use particularly for better connected clients. Using CDNs to serve bundles is also a good idea for well-connected clients. Compressed bundles can have a significant effect for all types of clients.
- In terms of lossy actions, removing embedded objects

TABLE VI
ATT.ICIR—ACTIONS WITH SIGNIFICANT IMPROVEMENTS OVER PARA-1.0 (+=20-50%, ++=>50%)

	Container Bytes-Embedded Objects-Embedded Bytes							
Action	S-S-S	S-M-M	M-M-M	M-M-L	M-L-L	L-M-M	L-L-M	L-L-L
para-1.0	0.69s	1.29s	1.29s	1.40s	2.82s	1.48s	2.05s	3.01s
pipe-1.1		+	+	+	++	+	++	++
cdn	+	++	++	++	++	+	++	++
bundle		+	+		++	+	+	++
bundle.gz		+	+	+	++	+	+	++
cdnbundle	+	++	++	++	++	++	++	++
baseonly	+	++	++	++	++	++	++	++
halfobject		+	+	+	+	+	+	+

TABLE VII
AU.WPI—ACTIONS WITH SIGNIFICANT IMPROVEMENTS OVER PARA-1.0 (+=20-50%, ++=>50%)

	Container Bytes-Embedded Objects-Embedded Bytes							
Action	S-S-S	S-M-M	M-M-M	M-M-L	M-L-L	L-M-M	L-L-M	L-L-L
para-1.0	1.46s	3.09s	3.61s	3.87s	7.00s	4.02s	5.65s	7.41s
pipe-1.1	+	++	+	+	++	+	++	++
cdn	+	++	++	++	++	++	++	++
bundle		+	+		++	+	+	+
bundle.gz		+	+	+	++	+	++	++
cdnbundle	+	++	++	++	++	++	++	++
baseonly	++	++	++	++	++	++	++	++
halfobject		+	+	+	+	+	+	+

has a significant effect in all cases. However, reducing the quality of embedded objects without reducing the number does not yield a significant improvement under most circumstances.

• Client connectivity, not latency, matters for determining which actions have significant performance effects. Our results are consistent for clients with similar connectivity despite large variations in latency.

In ongoing work, we are examining the cost to the clients for dealing with the modified content sent by the server. Amortization is feasible for clients behind a proxy but individual clients may have to absorb the cost for each modified response. In the case of decompression, most clients are already capable of handling it and prior work [5] has shown that decompression costs are insignificant. The costs of unbundling and other actions require additional investigation. We are also looking at other possible server actions that matter for repeat accesses for a pages (such as delta encoding) and policies regarding cachability of objects. We are currently incorporating our results in a Web server prototype that characterizes clients according to their connectivity and takes an appropriate action in the case of a client experiencing poor response.

REFERENCES

[1] Balachander Krishnamurthy, Craig Wills, and Yin Zhang. On the use and performance of content distribution networks. In *Proceedings of the ACM SIGCOMM Internet Measurement Workshop*, San Francisco, November 2001.

[2] Balachander Krishnamurthy and Craig E. Wills. Analyzing Factors that influence end-to-end Web performance. In *Proc. World Wide Web Conference*, May 2000.

[3] Balachander Krishnamurthy and Craig E. Wills. Improving Web Performance by Client Characterization Driven Server Adaptation. In *Proceedings of the World Wide Web Conference"*, May 2002.

[4] Media metrix, March 2002. `www.mediametrix.com`.

[5] Jeffrey C. Mogul, Fred Douglis, Anja Feldmann, and Balachander Krishnamurthy. Potential benefits of delta encoding and data compression for HTTP. In *Proc. ACM SIGCOMM*, August 1997.

[6] D. Mosberger and T. Jin. httperf—a tool for measuring web server performance. In *Proceedings of WISP '98, Madison, WI*, June 1998.

[7] Wget. `www.gnu.org/software/wget/wget.html`.

[8] Craig E. Wills, Mikhail Mikhailov, and Hao Shang. N for the price of 1: Bundling web objects for more efficient content delivery. In *Proceedings of the Tenth International World Wide Web Conference*, Hong Kong, May 2001.