



**The Association of System
Performance Professionals**

The **Computer Measurement Group**, commonly called **CMG**, is a not for profit, worldwide organization of data processing professionals committed to the measurement and management of computer systems. CMG members are primarily concerned with performance evaluation of existing systems to maximize performance (eg. response time, throughput, etc.) and with capacity management where planned enhancements to existing systems or the design of new systems are evaluated to find the necessary resources required to provide adequate performance at a reasonable cost.

This paper was originally published in the Proceedings of the Computer Measurement Group's 2000 International Conference.

For more information on CMG please visit <http://www.cmq.org>

Copyright 2000 by The Computer Measurement Group, Inc. All Rights Reserved

Published by The Computer Measurement Group, Inc., a non-profit Illinois membership corporation. Permission to reprint in whole or in any part may be granted for educational and scientific purposes upon written application to the Editor, CMG Headquarters, 151 Fries Mill Road, Suite 104, Turnersville, NJ 08012. Permission is hereby granted to CMG members to reproduce this publication in whole or in part solely for internal distribution with the member's organization provided the copyright notice above is set forth in full text on the title page of each item reproduced. The ideas and concepts set forth in this publication are solely those of the respective authors, and not of CMG, and CMG does not endorse, guarantee or otherwise certify any such ideas or concepts in any application or usage. Printed in the United States of America.

Web Page Design and Download Time

Jing Zhi

Keynote Systems

Statistical analysis of Web page download time measurements suggests that some relatively simple formulae can be derived to project page download times based on Web page composition and TCP connect time for a browser/server pair.

Introduction

Many factors contribute to Web site performance, most of which are at least partially outside the control of the site designer. Web page download times depend on page design, on Web server and client hardware and software configurations, and on the performance characteristics of the Internet route connecting a client to the site [Neil2000]. Of these, only page design is truly under the site designer's control.

However, if we assume a user with a high-speed connection and a site whose servers are not overloaded — as is typical of a business to business (“B2B”) interaction — then only two significant factors remain: site design and Internet latency between the client and the server [Sper1995] [Heid1997] [Touc1998]. In this paper we analyze measurement data based on test pages to explore various relationships between Web page design and page download time. Concentrating on information about the page and measures of Internet round trip time, we develop several specialized formulae to predict typical page download times in a B2B environment.

After some introductory discussion of Web download components and experimental setup, the first part of this paper identifies packet count, rather than page size, as the crucial predictor of download time. We indicate how to calculate packet count in the absence of packet sniffer software.

Next, we explore page download time as a function of page size, in a single-threaded environment. Here we build two different linear models to understand the bulk of page download for simple test pages (“Experiment A”).

Then, we investigate how multi-threading improves performance for more complex pages. It is particularly challenging to build accurate models for multi-threaded behavior since the distribution of download activity among threads can be altered substantially by a single Web page element that loads unusually slowly. A more complicated model is built and discussed for pages with one to 64 embedded images (“Experiment B”).

Since Internet performance varies erratically over time even for a fixed client-server pair, we discuss tradeoffs that can be made in modeling performance based on all available measurements vs. a more well-behaved

subset containing approximately 90% of available measurements.

We find that these experimental results represent and explain the basic mechanisms that regulate many of the performance characteristics observed while working with the extensive set of measurements collected annually by the author's company.

Components of Web Page Download Time

Before we can describe how we measure Web download performance, we include some background on the components that constitute Web page download. These will give us useful handles on different types of performance that we must build into our models.

Packets

On the Internet, all information is carried in packets. Network transfer times are not affected by the type of content being transmitted in those packets, but they are strongly influenced by the number of packets, and possibly even by their sizes. Also, the time required to set up a flow of packets is much larger than the amount of time between successive packets in a single connection.

The basic performance principle is therefore to make fewer requests and transmit fewer packets. From this principle, we can derive two basic design rules for well-performing Web pages. First, reduce the overall size of the page, thereby reducing the number of bytes (and packets) to be transferred over the Internet. Second, limit the number of embedded objects on the page, such as images, each of which must be requested and transferred separately from server to browser.

Web Page Download Components

To study Web page download time systematically, we consider its components, illustrated in Figure 1:

- DNS Lookup (DNS)
- TCP Connection (TCP)
- Redirection
- First Packet Download (FPD)
- Base Page Download (BPD)
- Content Download (CD)

COMPONENT	CLIENT	SERVER
DNS Lookup		
TCP Connection	Send SYN	
		Send SYN+ACK
	Receive SYN+ACK Send ACK	
Redirection	Send HTTP request	
		Send redirection response
First Packet Download	Send HTTP request	
		Perform server processing to locate or construct response
		Send HTTP response & first (or first two) HTML packets
	Receive 1st packet	
Base Page Download		Send remaining HTML packets ...
	Receive last HTML packet	
Content Download	Send HTTP GET requests	
	<i>TCP Connection, Redirection, First Packet Download, and Download phases for each</i>	
	Receive requested content	

Figure 1. Web page response time components

The first five components together determine how long it takes to load the HTML for a base page.

The First Packet Download is the time between the completion of the TCP connection with the destination server and the reception of the first HTML packet for the base page. It includes the client HTTP request and server HTTP response time. In addition to reflecting Internet latency, this measurement can also be an indicator of server performance and could differentiate between HTTP Performance and TCP stack speed.

The last component, called Content Download, is a little different from the first five. It lumps into one number the cumulative time for downloading all the embedded objects (such as images) on the page. This process may actually include *all five* of the previous component types, for *each* of the embedded Web objects. For example, one of our test pages includes 64 images, so the Content Download component for that page includes the total time required for DNS Lookup, TCP Connection, First Packet Download, and download of additional packets for all 64 images.

Although this description implies a huge overhead for each embedded element, in reality, many components of Content Download are either eliminated (by reusing cached DNS values and recycling existing TCP connections) or processed in parallel (by multithreading).

Modeling Challenges

This study investigates subsets of these components together with page design factors as predictors of total download time (including base page and content.) The

wild variability of the Internet makes it difficult to build such models.

For example, consider that one variable our model uses to predict total page download time is the TCP connect time measured for the base page only. In reality, the many different TCP connections obtained during the content download process might each perform quite differently. When such differences occur in practice, a single TCP time measured for the base page cannot tell the whole picture about the behavior of the TCP connections for multiple page elements.

Despite such difficulties, we found a good fit to modeled behavior.

Measurement Data

Each of the components described in the previous section was captured for downloads of the test pages described in the following sections. Downloads were taken from test computers (called “agents”) at three different city/backbone combinations:

- New York City / AT&T
- Houston / Qwest
- San Francisco / Sprint

Measurements were taken at five-minute intervals, so each page should have 288 measurements from each agent per day. In general, we have found that Internet performance differs remarkably not just over time, but also depending on the location and backbone from which content is requested. For this reason, we chose a variety of testing locations and used many repetitions of measurements to confirm patterns.

Agents have good Internet connectivity (a dedicated T1/T3). Except as described later, the agents run a proprietary Keynote browser designed to mimic the experience of business users.

Because this study focuses on the variability due to the Internet rather than that due to the server, the test pages and all embedded objects (image files) were hosted on the same server (www6.keynote.com).

This design decision has the side effect of eliminating the *redirection time* component from our study. To include redirection time in our results would require adding another variable to those we consider for each measurement. Since in practice the presence of a redirection component is an exception, rather than the norm, it is more practical to construct general models that do not include it, adding a separate estimate for redirection time only when it is applicable.

Pages Used for Experiment A

The first models focus on the effects of page size. We used pages with at most one embedded object. This

avoids multithreaded browser behavior. In these pages, we vary the size both of the base page and of the image, as shown in the table in Figure 2.

Pages Used for Experiment B

Next, we study multithreading as a mechanism to speed up downloads of complicated pages. We start with one of the pages used before (B.1 = A.7), and vary the number of pieces into which we chop up the same image.

As we will show, the benefits of multithreading are eventually outweighed as the number of images — and thus the likelihood of complications — increases.

The pages used in this experiment are described in the table in Figure 3. Each page uses the same total number of bytes (85,049), about 1K of which are in the base page (“HTML Size”). The remaining bytes are divided equally among multiple images in every page other than B.1, which contains a single image.

Page Index	HTML Size	# of Images	Total Content	Total Bytes	Web Page URL
A.1	62,300	0	0	62,300	www6.keynote.com/test/page/11.html
A.2	39,077	1	23,223	62,300	www6.keynote.com/test/page/12.html
A.3	20,893	1	41,407	62,300	www6.keynote.com/test/page/13.html
A.4	183	1	62,117	62,300	www6.keynote.com/test/page/14.html
A.5	183	1	18,118	18,301	www6.keynote.com/test/page/21.html
A.6	183	1	41,407	41,590	www6.keynote.com/test/page/22.html
A.7	1,136	1	83,913	85,049	www6.keynote.com/test/page/24.html
A.8	183	1	101,388	101,571	www6.keynote.com/test/page/25.html
A.9	183	1	120,665	120,848	www6.keynote.com/test/page/26.html

Figure 2: Test page information (Experiment A)

Page Index	HTML Size	# of Images	Total Content	Total Bytes	Web Page URL
B.1	1,136	1	83,913	85,049	www6.keynote.com/test/page/24.html
B.2	1,441	2	83,608	85,049	www6.keynote.com/test/page/32.html
B.4	1,293	4	83,756	85,049	www6.keynote.com/test/page/33.html
B.8	938	8	84,111	85,049	www6.keynote.com/test/page/34.html
B.16	1,049	16	84,000	85,049	www6.keynote.com/test/page/35.html
B.32	1,753	32	83,296	85,049	www6.keynote.com/test/page/36.html
B.64	2,233	64	82,816	85,049	www6.keynote.com/test/page/37.html

Figure 3: Test page information (Experiment B)

First Finding: Packet Count Is Much More Important Than Content Size

Before we begin to build the quantitative relationships found by experiments A and B, we cite a more general finding. For a given server-agent pair, the size (in bytes) of a page is a much poorer predictor of download time than the number of packets over which the page is transmitted. Because of this finding, we replace page size with packet count as an independent variable in our models.

Here, we summarize the finding and introduce the method to calculate the number of packets required for content of a given size.

Page Index	HTML	# of Images	Image Size	# of Packets	Total Content
B.64	2,233	64	1294	64x2	82,816
B.64.1	2,233	64	2262	64x2	144,768

Figure 4: Comparing packet count and page size

When applying a sniffer to the download of page B.64, we found that each image (of size 1294 bytes) was just large enough that together with its overhead, it did not fit into a single packet. As a result, 128 packets were required to send the 64 images. Another test page

B.64.1 was constructed with larger images (2262 bytes) that also fit into two packets. The two tests are summarized in Figure 4.

The histograms in Figure 5 show the measurement results for B.64 and B.64.1. These results show that, despite a 75% increase in content between B.64 and B.64.1, there is no substantial change in download time. Thus, even though downloading B.64.1 requires many *bigger packets* than downloading B.64, the fact that they can be downloaded in the *same number of packets* accounts for their similar download behavior.

Calculating Packet Count

For the sake of reference, we include a brief summary of how to calculate the packet count for given content. In our experiments, sniffers validated all packet counts.

The network packet size depends on the Maximum Transmission Unit (MTU), which is the greatest amount of data or packet size that can be transferred in one physical frame on the network.

Packet size is configurable for both servers and clients. On the network at speeds above 128Kbps, all connections are made with an MTU of 1500 bytes as default.

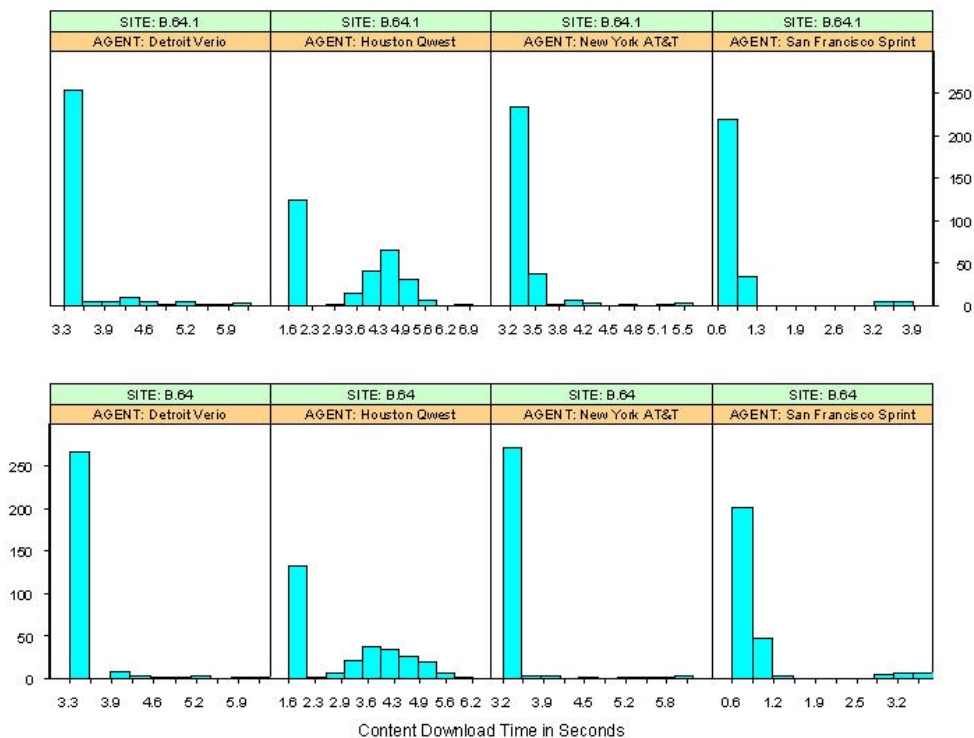


Figure 5: Content download times for sites B.64.1 and B.64

The Maximum Segment Size (MSS), the largest “chunk” size of data that TCP will send to the other end, would be 1460 bytes, subtracting 20-byte TCP header and 20-byte IP header from the MTU. Usually when a TCP connection is established, each end can announce its MSS. Agents for this study were homed at dedicated T1/T3 connections and used the usual MTU of 1500 bytes. Thus, they had TCP packet sizes of 1460 bytes.

The number of packets for an HTTP request may be calculated by the formula,

$$\# \text{Packet} = \lceil \text{Payload} / \text{MSS} \rceil \quad (1)$$

where $\lceil \]$ stands for rounding a number up to the nearest integer.

The *Payload* here is defined as the total number of data bytes required to transmit a Web page from the server through the Internet at the HTTP level, exclusive of TCP/IP header information. The payload also includes the HTTP response header, which is about 290 bytes for HTML pages.

Experiment A: The Effects of Total Download Bytes and Packets

Introduction

In this experiment, we investigate the dependence of download time on page size, as measured in packets. We will take two approaches to this problem. First, in **Model 1**, we perform separate linear regressions for each of three agents, examining results for their similarities and differences.

Given this experience, we construct a more general **Model 2** which uses the generic component of TCP connect time in a way that generalizes our analysis to Web clients located anywhere on the Internet. In each case, we mention novel results and provide additional validation in support of methodology and conclusions.

As a first finding, we observe that base page HTML and images download very similarly. Thus, we will refer to either as **payload**, and will investigate payload download as a function of packets.

For the download of either a base page or of an embedded object, two main phases may be distinguished. First, a connection is established (possibly after DNS lookup and redirection), culminating in transmission of a first “batch” of one or two packets. Next, a steady stream of packets begins flowing. The boundary we place between these phases derives partially from the fact that we measure at the client.

Whether we measure base page or embedded objects, for this experiment we seek to model the time taken for the second phase. We call this the **Payload Transmission Time (PTT.)**

Model 1: Individual Models Per Agent

To build the most basic model, we used test pages A1-A9. Note that two kinds of payload are considered: pages A1-A3 contain significant amounts of HTML, while pages A2-A9 include significant image content.

Measurements for this experiment were taken over one weekend day to minimize fluctuation in Internet latency, and to make the relationships as clear as possible.

Model 1: Measurement Results

To establish overall relationships, we based our model on the simple averages of the approximately 288 measurements per agent per target. Average performance per element is shown in Figure 6.

Consider the numbers of packets shown in Figure 6. These were determined using a packet sniffer. Since our Base Page Download component (the PTT in the case of HTML) does not include the first packet, the numbers shown for HTML are one less than the number of packets for the whole HTML page.

To obtain Payload Transmission Time (PTT) from the total content download time (CD), we subtract TCP and FPD of base HTML file from the total CD.

In Figure 7, we plot the Payload Transmission Time against the Number of Payload Packets for the three Keynote agents. On visual inspection, the relationships look very close to linear within each agent. This linear relationship is analyzed further in the discussion of Model 1.

Model 1: Statistical Model

We consider a basic linear least square regression model for each agent individually, where the Number of Packets (P) is the predictor (independent variable) and the Payload Transmission Time (PTT) is the response (dependent variable). We capture the different Internet connectivity for each agent into that agent’s regression coefficients, so that we avoid additional dependent variables related to Internet performance. Later models will take a more generalized approach.

To reduce the variance of the Payload Transmission Time, we use the arithmetic mean of PTT for each page as the response variable. According to the Law of Large Numbers, this averaging also allows us to treat the residual values as normally distributed. This assumption is quite reasonable even though individual Internet measurements are notoriously not normal.

The model for each agent can be stated as follows:

$$PTT = \mathbf{a} + \mathbf{b} \cdot P + \mathbf{e} \quad (2)$$

where \mathbf{e} is a random error term with mean $\mathbf{0}$ and variance \mathbf{s} ; and all the error terms are uncorrelated.

The fitted coefficients for each agent are listed in Figure 8.

Page Index	Content Type	Total Bytes	Observed #Packets	PTT in Seconds		
				Houston Qwest	New York AT&T	San Francisco Sprint
A.2	Image	23223	16	0.214	0.395	0.086
A.3	Image	41407	28	0.290	0.591	0.121
A.4	Image	62117	42	0.412	0.824	0.152
A.5	Image	18118	12	0.164	0.317	0.057
A.6	Image	41407	28	0.295	0.611	0.111
A.7	Image	83913	57	0.530	1.107	0.214
A.8	Image	101388	69	0.633	1.296	0.226
A.9	Image	120665	82	0.774	1.600	0.296
A.1	HTML	62300	42	0.384	0.796	0.144
A.2	HTML	39077	26	0.242	0.508	0.099
A.3	HTML	20893	14	0.142	0.307	0.058

Figure 6: Payload transmission time for individual elements

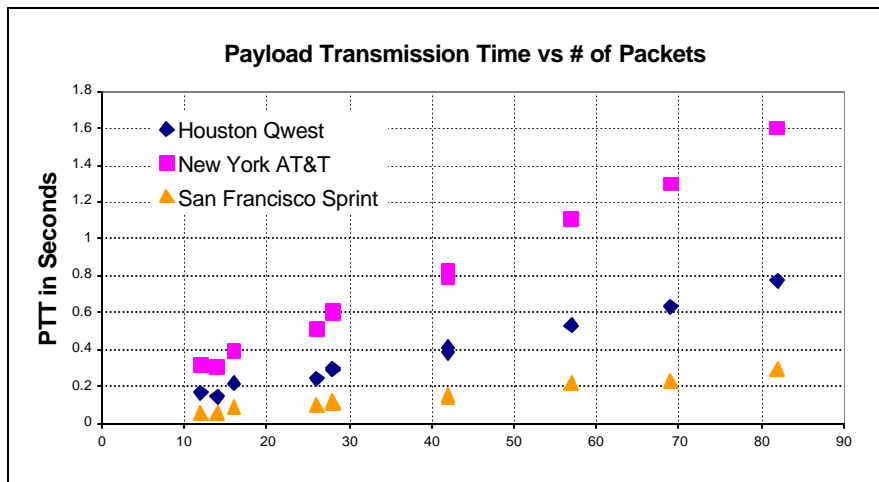


Figure 7: Payload transmission time vs. number of packets

	Houston Qwest	New York AT&T	SF Sprint
\hat{a}	0.0431	0.0748	0.0204
\hat{b}	0.0087	0.0181	0.0032
R^2	0.9909	0.9952	0.9815
TCP	0.0415	0.0936	0.0079

Figure 8: Fitted regression model coefficients and additional information

In Figure 8, R^2 , we also show the **coefficient of determination**, is an indicator of how well the data fits the least square model. It can be interpreted as the proportion of variability of the dependent variable that is explained by the independent predictor variable. The closer the R^2 is to 1, the better the linear association between the two variables is. We observe immediately that all the R^2 values are very close to 1, indicating very strong linear relationships (as would be expected based on a visual inspection of Figure 7).

Figure 8 also lists an average TCP round trip time, which is seen to correlate with both sets of fitted coefficients.

Model 1: Discussion

Observe how the coefficients \hat{b} are multiples of TCP time. This fits well with theories that try to explain download times as multiples of trips across the Internet. However, notice also that the approximate relationship

$$\hat{b} \approx 0.2 * TCP$$

of the other two agents fails to hold for the SF Sprint agent. Comparing the values of R^2 , we see that the model fits slightly less well for the SF Sprint than for the other two agents. A likely explanation is that, as measured by TCP round trip time, San Francisco Sprint is much closer than the other two agents to the target Web site server. When page download times are short, small variations in network conditions tend to have a relatively larger impact on overall page download times.

One added complication is that in the model above, the “intercept” terms \hat{a} of the regression models are very much nonzero. These are certainly statistically significant according to the t-test.

So what accounts for these terms? Theoretically, PTT should be zero when P goes to zero.

Because we have already factored out the TCP connect times and any server delays contained within the “First Packet Download” component, a positive intercept \hat{a} suggests another overhead factor is present.

A likely explanation for this is the *TCP slow start process* that applies at the beginning of packet flow. As described, for example, in [Stev1997], the first batch of packets in a TCP transmission may contain only one or two packets, after which the number of packets sent in each batch increases.

Alternately, network congestion may require that the packet transmission rate be adjusted downward.

In summary, it seems that our linear model's \hat{b} fits the steady state of flow that is achieved after slow start, while \hat{a} captures the overhead involved in ramping up to this steady-state packet flow. In Figure 9, the dotted lines illustrate our model, while the curved lines illustrate the underlying behavior.

It is interesting to see that just like the \hat{b} 's, the \hat{a} 's also appear related to the TCP round trip time. As with \hat{b} , the \hat{a} is a larger multiple TCP in the case of SF Sprint, where TCP is less. This suggests that in this case, \hat{a} is dominated by outlier “noise” that drives averages up.

The slope \hat{b} is interpreted as the increase in page download time in seconds per additional packet (P). In practice, both server and network conditions will affect the latency per packet transmitted. In this experiment, because all three agents measured the same Web server, we can reasonably expect the differences in the \hat{b} values to be related mainly to the differences in the round trip times between the agents and the target Web site servers.

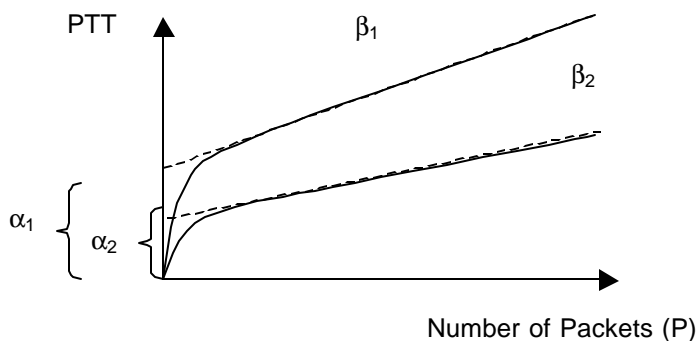


Figure 9: Relationship of the linear model and true behavior

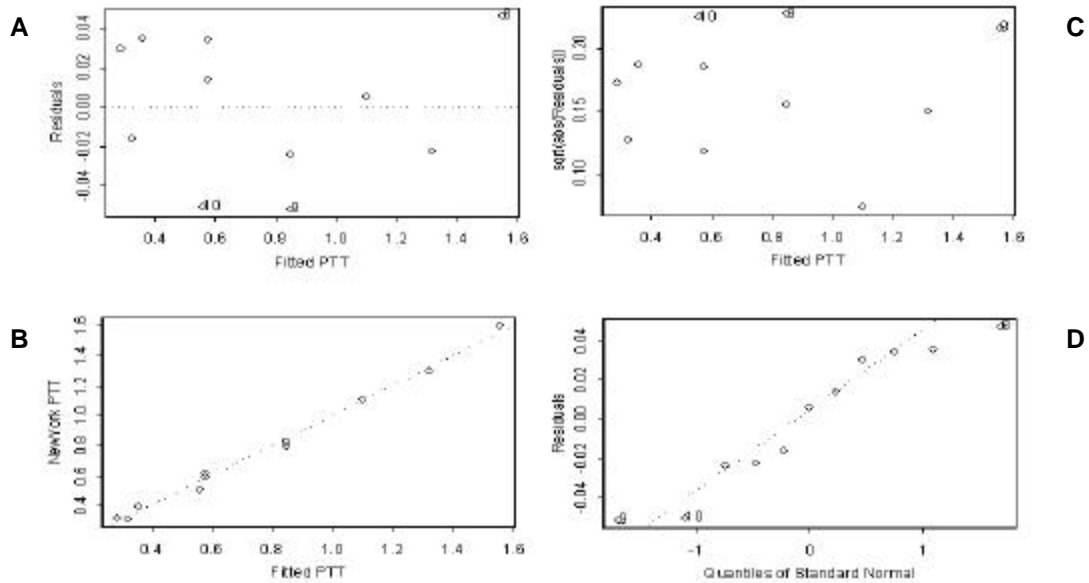


Figure 10: Diagnostics for Residuals

Additional Validation of Model 1

Under the least square model assumption, the residuals should look approximately like a sample of independent random normal noise with common variance, and with no pattern relative to the predictor or fitted PTT.

For the statistician, in Figure 10 we plot the diagnostic graphs of the fitted model residuals for the New York AT&T agent. We can see the linear regression model is basically valid. However, there is a slight curvature in the plots of residuals vs. fitted values (Figure 10: A & C). The curvature doesn't decrease much even if we try higher ordered regression models. While this demonstrates the usual pattern in Internet statistics of outliers behaving qualitatively differently than the majority of measurements, further investigation may uncover another cause.

Model 2: A Single Model For Multiple Agents Using TCP Time

From the previous model, we know of a reasonable predictor of PTT after \hat{a} s and \hat{b} s are fitted. Both of the parameters are related to the network round trip times. So in this section we generalize the previous model in two ways. First, we use the same coefficients for all agents, basing these on the common independent variable of TCP time. Second, we now fit to individual points, rather than to averages over pages.

The advantages of these generalizations are clear: not only is it easier to apply this model to measurements from other agents of other targets, but the model explains total time in terms of the easily understandable notion of a TCP round trip. Also, the model relates much more directly to user experience by studying individual times instead of averages.

However, this also raises significant challenges. Most notably, Internet statistics are tremendously variable, and the usual statistics (mean and especially standard deviation) are dominated by outliers. For this reason, we use least trimmed squares regression (LTS) [Burn1992], a highly robust method for fitting a linear regression model. Parameters fitted by LTS minimize the sum of certain fractions of smallest squared residuals.

The LTS approach has the effect of explaining the "well-behaved" majority of measurements, while not even considering the worst outliers. While it is unpleasant to have to dismiss outliers, in practice, one finds that models based on outliers are both unstable and usually a poor fit. In addition, linear models built using outliers tend to be so contorted as to obscure meaningful explanations of behavior.

Another difficulty arises from the fact that only one TCP time is measured per page, while different TCP round trips may behave very differently even over such a small time span as a single page download. This is part of the inaccuracy that must be accepted when building a model on so few variables.

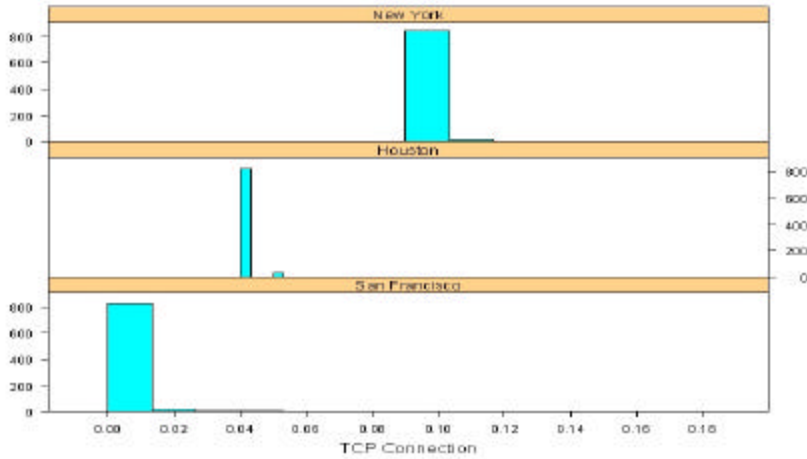


Figure 11: Histograms of TCP time by agent

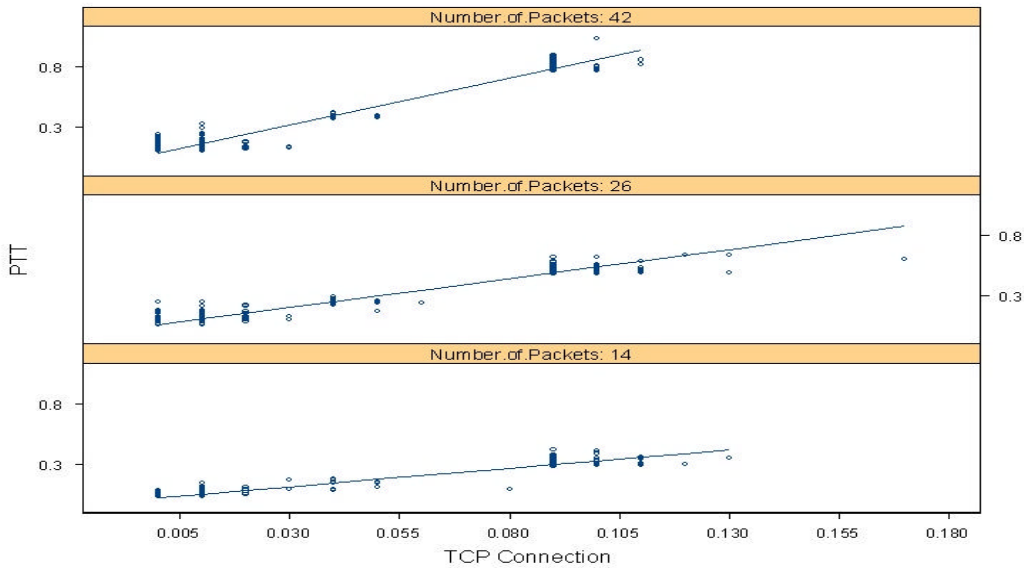


Figure 12: PTT as a function of TCP time, for different packet counts

Model 2: Measurement Results

The histogram in Figure 11 shows TCP Connection Time for each agent. As with most Internet component measurements, each histogram is far from normal, having a large right skew. In fact, a more detailed graph would show the histograms to be closer to lognormal, meaning that the logarithms of TCP times would appear close to normal. Again, to get around this difficulty, we use least trimmed squares regression.

We find that for a fixed number of packets, there is a very clear near-linear relationship between the PTT and TCP time. This is illustrated in Figure 12. The lines shown are the robust least trimmed square (LTS) regression fitting lines. Observe that Figure 12 mixes

measurements from different agents in each sub-graph. Some outliers fall outside of the displayed window. While it is hard to see here, the TCP time for each agent-target pair has a roughly lognormal distribution.

Model 2: Statistical Model

We build the linear model (lines in Figure 12) based on the HTML of test pages A1 – A3, as determined by the Base Page Download component. Independent variables are TCP time, the number of packets, and the interaction (i.e. product) of these two terms.

As before, the number of packets used here is the number of packets to transmit the base page minus one, to exclude the packet captured in the component First Packet Download.

The fitted linear model was determined as:

$$PTT = 0.4947 \cdot TCP + 0.0021 \cdot P + 0.1722 \cdot TCP \cdot P \quad (3)$$

The Robust Multiple R-Squared for this is 0.9962 and all the coefficients are statistically significant. About 90% observations determine the LTS estimates.

Corresponding to Model 1's term with coefficient \hat{b} , here we have the term including $TCP \times P$. Just as Model 1 had different values of \hat{a} and \hat{b} for each agent, here we must include more than one additional term: both the TCP and P terms together capture the effect of slow start, as well as the different balance between linear and constant terms seen for faster connections.

Model 2: Discussion

How can we interpret the coefficients in equation (3)? The coefficient for $TCP \times P$ is very straightforward. It models how efficiently the chunk of payload packets flows from server to client (agent). The P term could be related to delay at the server side in waiting for ACK packets before continuation of packet delivery.

The TCP term is somewhat consistent with how the intercept term α in Model 1 increases with the network round trip time (as varying from agent to agent.)

We should point out that this model may break down when the total payload is very small. This corresponds to the difference between Model 1 and reality

illustrated in Figure 9. Also, we emphasize that our "robust" fit excludes all those troublesome outliers, which while very different, are certainly not easy to ignore! In essence, we have modeled how things are "supposed" to work.

Further Validation of Model 2

In order to see how well the PTT could be predicted by this more general model, we took a further sample of measurements. Measurement agents were selected from both domestic and international locations, so that the measured TCP Connection Time components would vary a lot from agent to agent.

This sample also included measurements of a page that was not one of our original test pages, and that was served by another company's server at a different Internet location.

In this case, we used Base Page Download Time as a proxy for PTT. In Figure 13, we show sample data, predicted PTT, and the residuals (or observed minus predicted). The predicted times are calculated from the number of packets (determined by TCP packet sniffer or formula (1)) and measured TCP time.

In Figures 13 and 14 (which includes more data points than the Figure 13), the goodness of fit may be assessed informally by comparing the observed and predicted PTTs, which appear to agree quite well.

# of Packets	26	42	26	42	26	42	19	42	42	19	26	42
TCP	0.06	0.07	0.11	0.17	0.17	0.12	0.31	0.16	0.23	0.67	0.69	0.54
Observed PTT	0.33	0.58	0.6	1.01	1.02	1.06	1.15	1.41	1.72	2.46	3.58	4.45
Predicted PTT	0.35	0.63	0.60	1.40	0.90	1.02	1.21	1.32	1.87	2.56	3.49	4.26
Residual	-0.02	-0.05	0.00	-0.39	0.12	0.04	-0.06	0.09	-0.15	-0.10	0.09	0.19

Figure 13: Correspondence between modeled and observed times for typical sample data.

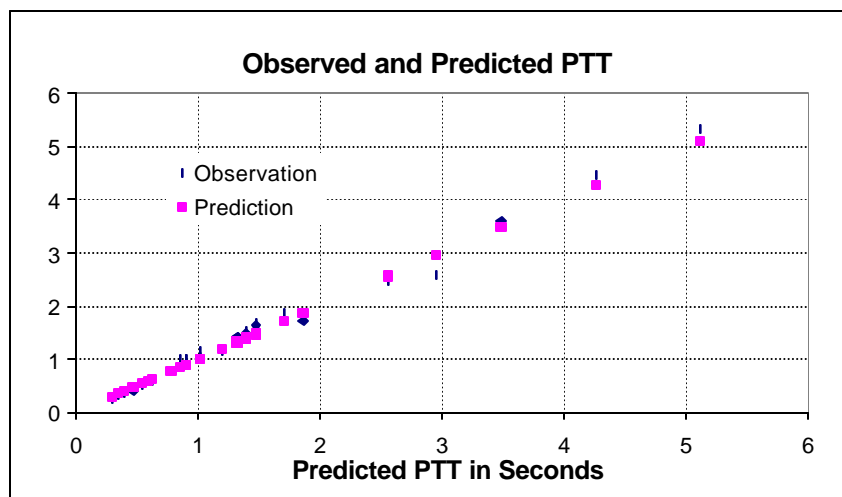


Figure 14: Graph of modeled and observed sample times

Experiment B: Number of Embedded Images

Introduction

Multithreading adds another layer of complexity into performance modeling. First, we show how delivery of the same image content varies depending on how many pieces it is sliced into, both in terms of performance and in terms of reliability. Next, we characterize the performance gain over single-threaded browsing as a quantitative gain in efficiency. We put all of this knowledge together in a model for multithreaded performance, which we follow with a discussion of tradeoffs between robust and regular least-squares modeling.

This section presents the most far-reaching results of the study and suggests directions for future work.

Multithreading

Most Web pages have more than one embedded image or program element. When downloading a Web page containing multiple embedded elements, browsers establish a separate TCP connection for each element, unless both client and server support persistent (“keep-alive”) connections, and the objects are located on the same server as the HTML index file or “base page” [HTTP1.0][HTTP1.1]. In our experiment, we assume the worst-case scenario, in which each embedded object requires the browser to open a new TCP connection.

When downloading Web page content elements, browsers can use multiple concurrent connections

(“threads”). For example, Internet Explorer uses 2 concurrent threads per server, and Netscape uses up to 6 (usually 4)[Wang1998]. The agents used in this testing use 4 concurrent threads. When the client (browser or measurement agent) can send multiple requests in parallel, and the server is able to accept those parallel connections, the total content download time is usually reduced. The actual reduction depends on the number of concurrent connections (“threads”) used, the bandwidth of the browser’s connection to the Internet, and the mix of element sizes being downloaded. While browsers connected via a dial-up connection will usually gain nothing from parallel threads (because the 56Kbs connection is a bottleneck), browsers connected via high-speed connections will benefit.

Measurements and Results

We fix our testing page total download sizes to 85,049 bytes in our experiment and change the number of embedded images. As test pages, we use the seven pages named B1 to B64, which have from 1 to 64 embedded images. The measured Content Download Time includes the time required (usually short) to decode the file after the HTML base page has been received, and the time to request and receive all referenced images.

Figure 15 shows the results. Note how performance improves with multithreading as one moves from one to four images, and then how performance worsens with more images as both the data transmission overhead and the probability of encountering problems increase.

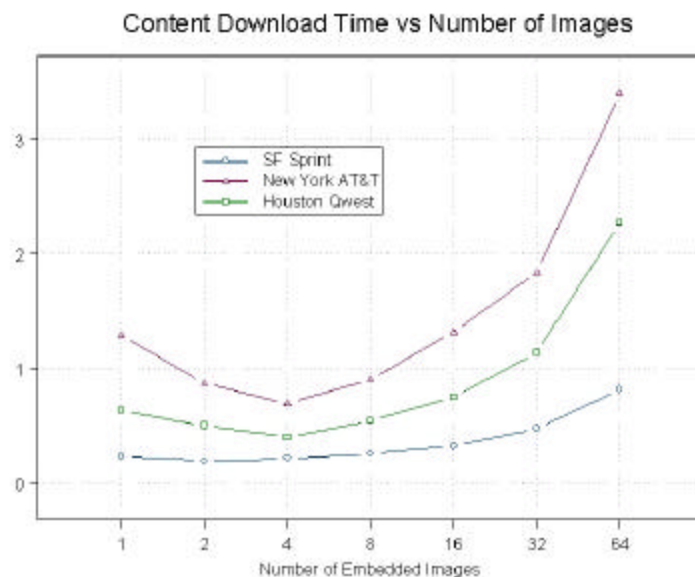


Figure 15: Content download time vs. number of images

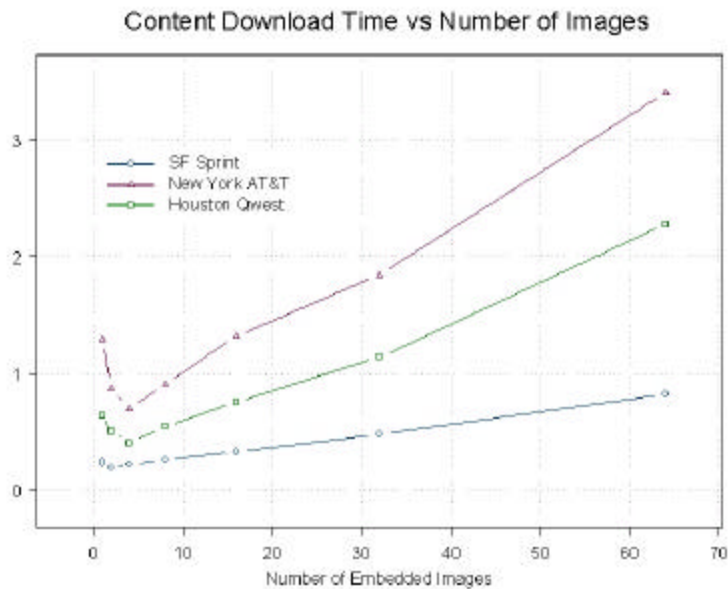


Figure 16: Content download time vs. linear scale number of images

To show quantitative relationships more clearly, Figure 16 illustrates the same data as Figure 15, except with the x-axis linear in the number of images, rather than on an exponential scale as in Figure 15.

The page with 4 images has the smallest mean download time according to the Houston Qwest or New York AT&T measurements. The total time to download the 64-image page increases dramatically. The download times almost double compared to the site with 32 images. In the case of New York AT&T, 1.55 additional seconds are required for such overhead as additional TCP/IP connections and slow starts.

Even if the client and server use persistent connections, the overhead of HTTP headers is an inevitable factor increasing download times. The HTTP header for each TCP/IP connection request is about 290 bytes, so the site with 64 images will have $290 \cdot 64 = 18,560$ additional bytes. More importantly, as explained in the “Finding 1” section above, this overhead necessitates additional packets to carry the images – a total of 128 rather than closer to 64 packets for the other pages.

Errors

We also tracked errors over a continuous week of measurements; the results are shown in Figure 17. We observe that content errors increase roughly linearly with the number of images. These errors occur when the base page loads correctly, but one or more of the content elements fail. This could be due to TCP connection failure, for example.

As page complexity increases (for example with a larger number of images), a connection time out error becomes more likely. Although the association of time outs with image count is not so clear here, in general,

this should be a concern. Other types of errors shown occurred during base page download, and bear no apparent relationship to the number of images.

Number of Images	1	2	4	8	16	32	64
Connection Refused	0	0	0	0	1	0	0
Connection Timed Out	2	2	2	5	3	4	2
Content Error	0	4	8	13	39	67	114
Unknown Error	0	0	3	0	0	0	0
Total Errors	2	6	13	18	43	71	116

Figure 17: One-week table of measurement errors

In our general experience, the occurrence of errors is strongly associated with the occurrence of outliers, which in turn associate with generally poor performance. Thus, before we even begin the quantitative analysis, we can draw important conclusions for page design.

When attempting to optimize performance of a page with multiple elements, be sure to include enough elements so as to keep all browser threads occupied. Ideally, no single element should be so much larger than others that it forces its thread to take longer than other threads. However, when the number of elements per server far exceeds the number of concurrent threads supported by the browser, additional elements increase overhead, and slow down the overall download time.

# of Images	Estimated Content Packets, based on size	Predicted Content Download Time (No Concurrent Connection)		
		Houston Qwest	New York AT&T	San Francisco Sprint
1	57	0.64	1.29	0.22
2	56	0.73	1.46	0.24
4	56	0.93	1.89	0.30
8	56	1.25	2.57	0.34
16	48	2.06	3.96	0.45
32	32	3.54	6.78	0.71
64	64	7.01	13.55	1.60

Figure 18: Predicted Content Download Time With No Concurrent Connection.

Concurrent Connection Efficiency

The content download time with no concurrent connection can be predicted by the formula:

$$n (TCP + FPD) + PTT \quad (4)$$

where the *PTT* is predicted from the adjusted total content download packets and *n* is the number of embedded images. The predicted values for the three agents are illustrated in Figure 18:

Obviously, the predicted content download times with no concurrent connection are much longer than actual measurements. But how much benefit do the pages gain from client multithreading?

We define the **Concurrent Connection Efficiency** as the ratio of Content Download Time with no Concurrent Connections (predicted) over Content Download Time with Concurrent Connections (measured). The perfect efficiency would be lesser of *n* and the number of concurrent connections (4 in these tests).

The images in the pages with 16, 32 and 64 images are equal sized. According to the data in the figure 19, we could see that these three pages benefit more from the concurrent connection than other pages. New York AT&T, with a longer Internet round trip time, achieves more Concurrent Connection Efficiency than the other two. San Francisco Sprint, with smallest latency, benefits least from the multithreading. One way this

might possibly be explained is that the connection between the agent and the server might have been so good that the server itself became a larger part of the bottleneck. In this case, multithreading at the client does not speed up the server's delivery of packets as much.

Models for Multithreaded Performance

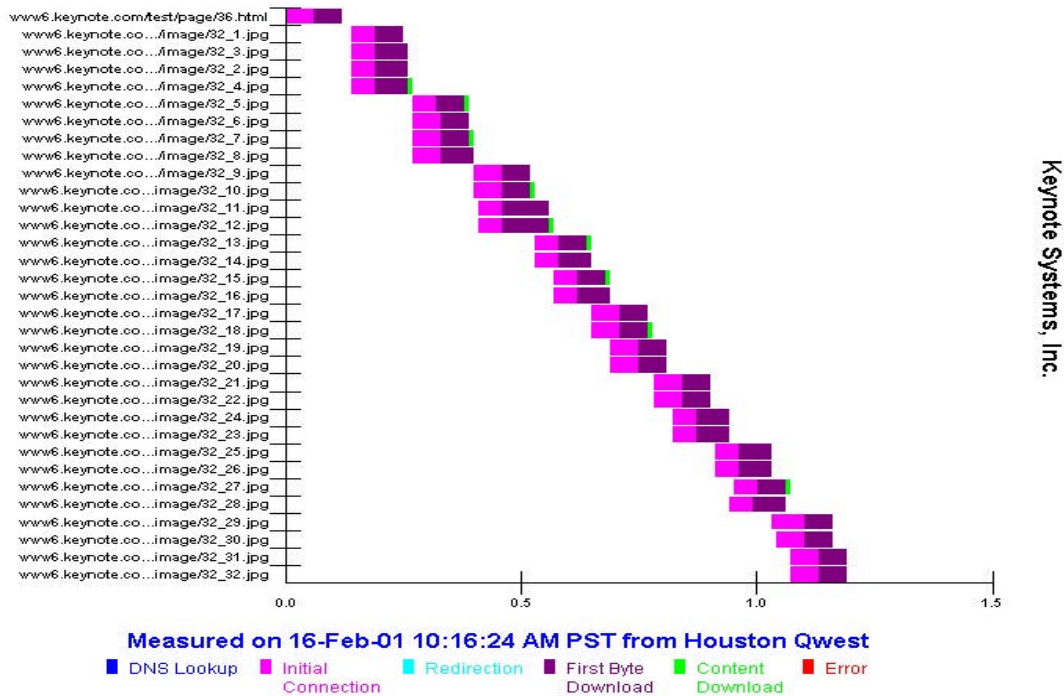
As more than one embedded image is added to the base page, the Internet download time becomes much more complicated. This is illustrated in detail in Figure 20, which is a graph of a download of test page B.32. The components of download time are coded using different shades of gray. For each element, most time is spent either forming TCP connections or waiting for the first packet.

In this case, the browser uses 4 parallel threads. New threads are spawned for individual images only after the complete base page has been received and parsed. In addition, individual connections are established for each object. Many of these behaviors resemble Netscape more closely than Microsoft's Internet Explorer. A subsequent paper will build similar performance models more closely tailored to IE.

# of Images	Content Download Time (4 Thread Connection)			Concurrent Connection Efficiency		
	Houston Qwest	New York AT&T	SF Sprint	Houston Qwest	New York AT&T	SF Sprint
1	0.63	1.29	0.24	1.01	1.00	0.95
2	0.50	0.87	0.20	1.45	1.68	1.24
4	0.40	0.70	0.22	2.30	2.71	1.36
8	0.54	0.91	0.25	2.30	2.83	1.35
16	0.75	1.32	0.33	2.74	3.01	1.38
32	1.14	1.84	0.48	3.12	3.68	1.50
64	2.27	3.40	0.82	3.08	3.98	1.95

Figure 19: Content Download Time Observations and Concurrent Connection Efficiency

www6.keynote.com/test/page/36.html Full Page Measurement



Keynote Systems, Inc.

Figure 20: Detailed download performance for test page B.32 (i.e. with 32 images)

Notice in Figure 20 that despite the identical size and packet counts of the images, groups of four images processed simultaneously tend to drift apart. As a result, it is quite common for some threads to process more images than others. This is only one of the difficulties in modeling multithreaded content download.

We may model the total content download time (CD) in terms of a single thread with the maximal workload, basing our model on the table in Figure 21. The abbreviated variable names P/O, P, and O shown in this table will be used later in our derived model formulae. Note that when there are fewer than

4 images, the number of threads used equals the number of images.

To build this model, we make two small simplifications. First, we treat the variable *Images per Thread* (abbreviated "O" for Objects) as if each thread handled the same number of images. In reality, a single slow image may delay its thread so much that other threads may end up having to handle more than their "fair quota" of images. Second, we are modeling the maximum time of a known number of threads in a linear model, even though taking a maximum (even of several linear variables) is not a linear operation.

Page Index	# of Images	Maximum # of Packets per Image (P/O)	Maximum # of Packets per Thread (P)	# of Images per Thread (O)	# of Threads in Use
B.1	1	58	58	1	1
B.2	2	30	30	1	2
B.4	4	17	17	1	4
B.8	8	9	18	2	4
B.16	16	4	16	4	4
B.32	32	2	16	8	4
B.64	64	2	32	16	4

Figure 21: Test page characteristics for Experiment B

In fact, the fitting of the model corrects for these simplifications implicitly. First, for example, for page B.8, we might find that the average number of images handled by the slowest thread was 2.1, rather than the 2 in our model. If so, the ratio 2.1/2 would be included in the fitted linear coefficient for *Images per Thread*. Second, in this same example, the expected time for the slowest of the four threads for downloads of B.8 would be strictly greater than the expected time of individual threads handling 2 images. Again, this overhead would be included in coefficients fitted to the model.

So we emphasize that while the model uses predictor variables that are based on simplifying assumptions, the fitting of the model uses real measurements, and so counterbalances the assumptions. Thus, as we see below, the predicted *total* download time behaves quite accurately. On the other hand, it is not appropriate to interpret the fitted model coefficients as modeling the expected download time per image for an *average* thread, for example.

The Generalized Model

Content Download Time (CD) can be modeled as:

$$CD \propto TCP \cdot O + FPD \cdot O + f(P/O) \cdot O \quad (5)$$

This notation means that we will fit constants to multiply by each of the three terms listed on the right side of the formula to model Content Download Time.

The first two terms are related to the overhead of TCP connection and server processing time to download each image. As before, we do not obtain TCP and FPD times for each image separately, but rather use those times measured for the base page of a download. As is seen clearly in Figure 20 above, both of these components vary across elements, but this way, we are able to build a simpler, though somewhat less precise, model.

The last term is a content data transmission time, such as was the main result of Model 2. There we pointed out the use for possible additional terms, such as intercept and non-interaction terms. However as a start, we simplify $f(P/O)$ to a linear function, allowing us to rewrite the previous model as

$$CD \propto TCP \cdot O + FPD \cdot O + TCP \cdot P + O \quad (6)$$

Here we have also replaced the term $(P/O) \times O$ by P , indicating the amount of packets traveling within a thread. Also, we have incorporated the multiplier TCP into this term, to account for the Internet round-trips needed to schedule additional packet flow.

Finally, we have added a separate O term in the way that interaction terms in Model 2 also used single-variable terms.

By looking at Figure 15, the Page B.4 with a relatively small number of Packets per Thread (P) and the least number of images per thread (O) needs shortest response time to download the content. This agrees with formula (6).

As in Model 2, we model the content download time by using LTS robust regression to fit the coefficients of terms in (6) based on the individual observations of the B.1-B.64 Web pages on one weekend measured by agents at the usual three locations. LTS results are shown in equation (7).

Discussion

R-Squared is close to 1 after 10% of the observations are thrown away. As in the earlier discussion, this means that the majority (~90%) of measurements behave in an easily understandable way.

The intercept in the model could be the time the client parses the HTML after downloading the whole HTML text file. Alternately, it could just fill in for some of the random fluctuation that is not explained by other coefficients.

The coefficient of $TCP \cdot P$ is pretty close to the one in Model 2. Theoretically, the true coefficients of $TCP \cdot O$ and $FPD \cdot O$ should be 1, since these cover the same tasks that need to be completed either when sending an image or the base page. However, the fitted coefficients are slightly less than 1. This suggests some obscuring of individual time contributions such as by multithreading.

In this linear model, all of the TCP slow start would be counted in term O, which certainly has a nonzero coefficient.

Thus, we find that all of the results seem have sensible interpretations. Note that other regressions were attempted using additional variables (such as the number of threads in use), but that these models did not come out as significantly more accurate than the results already discussed.

Later work will concentrate on more general page designs. For real Web pages, a difficulty in applying this model is that each embedded page element can have a different size, so that it is not clear which subset of elements will be handled by the slowest thread. Nevertheless, we anticipate that substantial progress can be made even in the multithreaded cases.

$$CD = 0.1273 + 0.8684 \cdot TCP \cdot O + 0.9182 \cdot FPD \cdot O + 0.1939 \cdot TCP \cdot P + 0.0135 \cdot O \quad (7)$$

Scale estimate of residuals: 0.07151

Robust Multiple R-Squared: 0.9782

Total number of observations: 6044

Observations used in LTS regression: 5439.

Results of Non-Robust Regression

The results discussed in the previous section were only the most clear of several attempts. Originally, a larger combination of predictor terms was used, and less meaningful (and less significant) terms were omitted to produce the previous results.

A different approach attempted was to build a model using linear regression on *all* of the data points. As bad as outliers are in linear statistics such as the mean, they can wreak much greater havoc when given even greater weight, such as in least *squares* regression. While we will not even publish the coefficients returned by regression (for fear of misleading the reader), suffice it to say that the R^2 for the full fit was only about 0.66.

So in summary, the roughly 10% of measurements omitted from the robust regression not only brought the R^2 down over ten times as far from perfect, but they managed to distort wildly the regression model. It is clear that any model that seeks to avoid trimming must use a different technique than least squares regression of linear data. However, since the linear model fits most of the data so well, and since it has such an intuitively appealing interpretation, the most attractive approach is to study the outliers as a sub-population, and in fact as several sub-populations, if warranted.

Conclusion

This paper represents part of an ongoing effort to understand better the billions of measurements taken annually by Keynote Systems. Despite the complexities of dealing with heavy-tailed Internet statistics and such technologies as multithreading, we feel that the results discussed here show significant promise toward building more granular quantitative performance models for entire Web pages based solely on page design and simple measures of Internet latency as taken only for a base page. Subsequent work should generalize these results to more kinds of pages as well as other browser technologies.

Implications useful for site designers include the goal of keeping all available threads “busy” for the same amount of time. For example, this might be improved by separating a page’s single largest image into parallelizable pieces.

On the other hand, our performance optimum at four images is dependent on equal image sizing as well as on a lack of connection recycling. If a browser keeps connections to a server open for possible additional element downloads, this would tend to tilt upward the number of images that optimize performance. In any case, large numbers of images cause undesirable overhead, and image sizes should be understood primarily in terms of the number of packets they require.

Acknowledgement

I am indebted to Chris Overton for his expert guidance in the experiment design and in the research work of the field, and for his assistance with the creation of this paper. I would also like to thank Shawn White for his assistance with the measurement setup for this work, and Dr. Jeff Buzen, Richard Gimarc, and Chris Loosley for helpful comments and suggestions on earlier drafts of this paper.

References

- [Burn1992] Burns, P.J, A Genetic Algorithm for Robust Regression Estimation. (Statsci Technical Note)
- [Heid1997] John Heidemann, Katia Obraczka, and Joe Touch. Modeling the Performance of HTTP Over Several Transport Protocols. ACM/IEEE Transactions on Networking, 5 5, 616-630, October, 1997
- [HTTP1.0] Hypertext Transfer Protocol HTTP/1.0 Specifications 1998
- [HTTP1.1] Hypertext Transfer Protocol HTTP/1.1 Specifications
- [Neil2000] Jakob Nielsen, Designing Web Usability: The Practice of Simplicity, New Riders Publishing 2000
- [Sper1995] Simon E Spero Analysis of HTTP Performance problems
<http://www.ibiblio.org/mdma-release/http-prob.html>
- [Stev1997] W. Richard Stevens; TCP/IP Illustrated, Volume 1: The Protocols (Addison-Wesley)
- [Touc1998] Joe Touch, John Heidemann, and Katia Obraczka. Analysis of HTTP Performance. Research Report 98-463, USC/Information Sciences Institute, August, 1998
<http://www.isi.edu/lam/publications/http-perf/>.
- [Wang1998] Zhe Wang and Pei Cao, “Persistent Connection Behavior of Popular Browsers”
<http://www.cs.wisc.edu/~cao/papers/persistent-connection.html>