Understanding Website Complexity: Measurements, Metrics, and Implications

Michael Butkiewicz
UC Riverside
butkiewm@cs.ucr.edu

Harsha V. Madhyastha
UC Riverside
harsha@cs.ucr.edu

Vyas Sekar
Intel Labs
vyas.sekar@intel.com

ABSTRACT

Over the years, the web has evolved from simple text content from one server to a complex ecosystem with different types of content from servers spread across several administrative domains. There is anecdotal evidence of users being frustrated with high page load times or when obscure scripts cause their browser windows to freeze. Because page load times are known to directly impact user satisfaction, providers would like to understand if and how the complexity of their websites affects the user experience.

While there is an extensive literature on measuring web graphs, website popularity, and the nature of web traffic, there has been little work in understanding how complex individual websites are, and how this complexity impacts the clients' experience. This paper is a first step to address this gap. To this end, we identify a set of metrics to characterize the complexity of websites both at a content-level (e.g., number and size of images) and service-level (e.g., number of servers/origins).

We find that the distributions of these metrics are largely independent of a website's popularity rank. However, some categories (e.g., *News*) are more complex than others. More than 60% of websites have content from at least 5 non-origin sources and these contribute more than 35% of the bytes downloaded. In addition, we analyze which metrics are most critical for predicting page render and load times and find that the number of objects requested is the most important factor. With respect to variability in load times, however, we find that the number of servers is the best indicator.

Categories and Subject Descriptors

D.2.8 [**Metrics**]: [Complexity measures]; D.4.8 [**Performance**]: [Modeling and prediction]

General Terms

Measurement, Human Factors, Performance

Keywords

Web page complexity, Page load times

IMC'11, November 2–4, 2011, Berlin, Germany.

1. INTRODUCTION

Over the last decade, web pages have become significantly more complex. Originally used to host text and images, web pages now include several content types, ranging from videos to scripts executed on the client's device to "rich" media such as Flash and Silverlight. Further, a website today fetches content not only from servers hosted by its providers but also from a range of third party services such as advertising agencies, content distribution networks (CDNs), and analytics services. In combination, rendering a single web page today involves fetching several objects with varying characteristics from multiple servers under different administrative domains.

On the other hand, the ill-effects of slow websites are well-documented. Recent surveys suggest two thirds of users encounter slow websites every week [15] and that 49% of users will abandon a site or switch to a competitor after experiencing performance issues [9]. While there is plenty of anecdotal evidence that the increase in web page complexity is a key factor in slowing down websites, formal studies on this topic have been limited. Most prior work on web measurement focuses on characterizing the Web graph [19, 16], analyzing the network footprint of Web traffic [34, 35, 41, 39, 42, 30], or studying the rate of change of content on the Web [29]. While these have contributed to a better understanding of web usage, they do not analyze the websites themselves.

In this paper, we present a comprehensive measurement-driven study of the complexity of web pages today and its impact on performance. We measure roughly 1700 websites from four geographically distributed locations over a 7 week period. These websites are spread across both a wide range of popularity ranks and genre of website categories. In analyzing website complexity, we focus on a *client-side* view of the landing pages of these sites and not on the dependencies in the back-end server infrastructure—an interesting topic complementary to our efforts.

Understanding the complexity of web pages and its implications is vital on several fronts. With the increasing diversity of client platforms for accessing the Web, it is important for browser developers to identify the aspects of web page complexity that impact user-perceived performance. On the other hand, as website providers increasingly incorporate third-party services such as advertising, analytics, and CDNs into their webpages, they need tools and techniques to evaluate the impact of these services on users. Furthermore, beyond the perspective of any given user or web provider, understanding website complexity is a first step toward solutions for automatically customizing web pages for varying client platforms to achieve the right balance between performance, usability, and business interests.

Our study focuses on two broad questions. First, we quantify the complexity of a web page with a broad spectrum of metrics.

We characterize a web page by the content fetched in rendering it—the number of objects fetched, the sizes of these objects, and the types of content. While these features remain largely the same across different rank ranges of websites, we see a marked difference across different website categories. For example, *News* websites load a significantly higher number of objects than others, whereas *Kids and Teens* websites host a higher fraction of Flash content.

In addition to characterizing this content-level complexity, we study the complexity of web pages with respect to the services they build upon. We find that non-origin content accounts for a significant fraction of the number of objects and number of bytes fetched, an observation that holds even on low ranked websites. However, the impact on download time of non-origin content is low—the median contribution to download time is only 15%. Though the most popular third-party services are unsurprisingly analytics and advertising providers, emerging services such as social networking plugins and programming frameworks also appear on a sizeable fraction of websites. A significant difference that we observe in the types of content served from non-origins in comparison to that from website providers themselves is that Javascripts account for a much higher fraction of non-origin objects.

The second focus of our study is to identify the critical complexity metrics that have the most impact on the time to download and render a web page. We find that rather than the total number of bytes fetched to render a website, the number of objects fetched is the most dominant indicator of client-perceived load times. We corroborate this with a linear regression model that predicts page load times with a normalized mean squared error less than 0.1. We also determine that, in contrast to actual load times, variability in load times is better correlated with the number of servers from which content is fetched.

2. RELATED WORK

There have been many efforts to analyze different aspects of the Web ecosystem. This includes work to understand web structure, tools to improve web performance, and measurements of emerging web applications. We describe these next. Note that most of these efforts focus either on *web traffic* or *web protocols*. There has been surprisingly little work on quantifying and understanding *website complexity*.

Structure and evolution: The literature on modeling the Web graph and its evolution focus on the interconnecting links between websites [19, 16] rather than the structure and content of individual websites. Related efforts have studied how the content of individual web pages evolves over time [29]. Recent efforts have also tried to "map" the hosting sites from which content is served [20].

Performance and optimization: As the usage scenarios for the Web have changed, researchers have analyzed inefficiencies in web protocols and suggested improvements [38, 21, 11]. In parallel, there are efforts toward developing better browsers [37], tools to optimize webpages [2, 1, 13], benchmarking tools [27, 12, 24], services for customizing web pages for different platforms [40, 10, 8], and techniques to diagnose performance bottlenecks in backend infrastructures [43] and to debug client performance in the wild [32].

Web traffic measurement: This includes work on measuring CDNs [34], understanding emerging Web 2.0 and AJAX-based applications [35, 41], measuring the network impact of social network applications [39, 42], and characterizing end-user behavior within enterprises [30], and longitudinal studies [31] among many others. These focus on web traffic as observed at the network-level, and not on understanding the structure and performance of individual websites.

Impact of load time on users: Several user experience studies evaluate how page load times impact user satisfaction [18, 26]. There are also commercial services that measure page load times in the wild [5]. These highlight the importance of optimizing page load times. However, there have been few attempts to understand how different aspects of website complexity impact the time to load web pages.

Privacy leakage: Krishnamurthy et al. [33] report the proliferation of third-party services. Our measurement setup is similar to theirs and we quantify the use of third-party services as well. However, our end goals are very different. In particular, they focus on the privacy implications and observe that a small number of administrative entities (e.g., Google, Microsoft) have broad insights into web access patterns. Our focus, instead is on using the presence of third-party services as a metric to characterize website complexity and on studying their impact on page load times.

Complexity metrics in other domains: Other efforts present metrics to quantify the complexity of network protocols [25], network management [22], and systems more broadly [23]. In a Web context, Zhang et al. [44] present metrics to capture ease of web page navigation and Levering et al. [36] analyze the document layout structure of web pages. The high-level motivation in these efforts is the need for quantitative metrics to measure system complexity and understand its impact on performance and usability. Our study follows in the spirit of these prior efforts to quantify website complexity to understand its impact on page load times.

Characterizing webpages: The closest related work appears in recent industry efforts: HTTP Archive [3] and at Google [6]. While the data collection steps are similar, we extend their analysis in two significant ways. First, we consider a more comprehensive set of complexity metrics and present a breakdown across different rank ranges and categories. Second, and more importantly, we go a step further and construct models for correlating and predicting performance and variability in performance with respect to the measured complexity metrics. Furthermore, we view the presence and timing of these parallel industry efforts as further confirmation that there is a *key gap* in understanding website complexity and its performance implications. Our work is a step toward addressing this gap.

3. MEASUREMENT SETUP

We begin by describing the measurements we gathered that serve as input for all of our analysis. All our datasets can be downloaded at `http://www.cs.ucr.edu/~harsha/web_complexity/`. We start with around 2000 websites at random from the top-20000 sites in Quantcast's list of most popular websites.[1] We annotate these sites with content categories obtained from Alexa.[2]

To faithfully replicate the actions of a web browser when a user visits a website, we use a browser (`Firefox`) based measurement infrastructure. We use a "clean" Firefox instance (version 3.6.15) without any ad or pop-up blockers. (We install suitable plugins such as the Adobe Flash player to ensure that websites render properly.) We use the `Firebug` extension (version 1.7X.0b1) with the `Net:Export` (version 0.8b10) and `Firestarter` (version 0.1.a5) add-ons to automatically export a log of all the requests and responses involved in rendering a web page. This extension generates a report in the HTTP archive record (HAR) format [4] that provides a detailed record of the actions performed by the browser in loading the page.

[1] `http://www.quantcast.com/top-sites`
[2] `http://www.alexa.com/topsites/category`

```
"log":{
  "version":"1.1",
  "browser":{
    "name":"Firefox",
    "version":"3.6.11"
  },
  "pages":[{
      "startedDateTime":"18:12:59.702-04:00",
      "title":"Wired News",
        "pageTimings":{
          "onContentLoad":2130,
          "onLoad":4630}]
  "entries":[{
        "startedDateTime":"18:12:59.702-04:00",
        "time":9,
        "request":{
          ...
  "headers":[{
              "name":"Host",
              "value":"www.wired.com" },
         ...}
        "response":{
         ...
         "content":{
           "size":186013,
           "mimeType":"text/html",}},
       }]
       ]
```

Figure 1: Example snippet showing the different fields in a HAR file. From this report, we can reconstruct the exact sequence of requests and responses, the number and size of requests for objects of different content types, and identify the server/origin for each request.

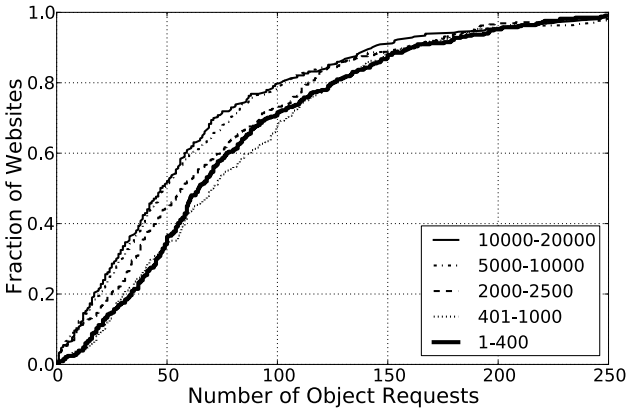
Rank range	Number of websites
1-400	277
400-1000	298
2000-2500	330
5000-10000	443
10000-20000	400
All	*1748*

Table 1: Summary of spread across rank ranges of websites in our measurement dataset.

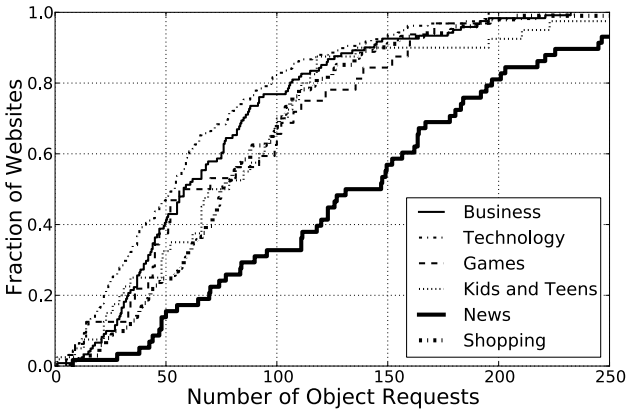
Figure 1 shows a snippet of a HAR report. It reports two page load metrics—`onContentLoad`, which is the time taken to start rendering content (we call this RenderStart), and `onLoad`, the time to completely render all components of the page (we call this RenderEnd).[3] Our main field of interest is the array of request-response entries. Each entry provides a timestamp of when the request was initiated, the time it took to complete the request, the host to which the request was sent, and the size and content type of the response.

We gathered measurements from four geographically distributed vantage points. Three of these vantage points were Amazon EC2 Micro instances running Ubuntu Linux (version 11.04) located in the US-East, Europe, and Asia Pacific regions of EC2. To discount any effects specific to EC2, our last vantage point is a personal desktop at UC Riverside. We choose multiple vantage points to ensure that our choice of measurement site does not introduce any bias.

[3]There is discussion in the web community on good load time metrics; suggestions include "above-the-fold" time, time to first "paint", time-to-first-byte, etc. [7]. Using a full spectrum of render metrics is outside the scope of our analysis. We pick the two standard metrics reported by the browser itself.



(a) By rank



(b) By category

Figure 2: Total number of objects loaded on the base web page of websites across rank ranges and categories.

At each vantage point, we run a measurement agent that periodically (every 60 seconds) selects a website at random from the list of 2000 sites, launches a Firefox instance that loads the base URL (i.e., `www.foo.com`) for the website, generates the log report in the HAR format, and subsequently terminates the Firefox process. We repeat this for a period of 9 weeks (between May and August 2011) and gather roughly 30 measurements per website on average. Our primary focus is on the root or landing pages of these sites; we present a preliminary study of non-landing pages in Section 6.

We perform the following pre-processing on our data. We discard individual HAR files if they have recorded the number of bytes fetched or the page load time as zero, specify the HTTP status code as not 200, or are malformed. We discard measurements from all websites for which we observe consecutive HAR files spaced less than 60 seconds apart. These correspond to corner cases where the Firebug add-on to Firefox had trouble exporting a HAR file for the web page it loaded. These steps for cleaning the dataset leave us with measurements from 1748 websites. Table 1 summarizes the spread of the websites that remain across five rank-ranges.

For each website, we compute the median value for various features, such as number of objects loaded or number of servers contacted, across multiple measurements of the website. We use these median values of features for most of our analysis, though we do

Complexity metric	Key result(s)
Content complexity	
No. of objects	Across all rank ranges, median web page requests over 40 objects and 20% request over 100 objects *News* websites load a significantly greater number of objects than others
MIME type	Contribution of various content types is similar across rank ranges Images dominate in fraction of objects, but to a lesser extent with respect to fraction of bytes *Kids and Teens* websites have a significantly greater fraction of Flash content than others
Service complexity	
No. of servers	25–55% of websites load content from at least 10 servers *News* websites fetch content from significantly more servers than others
Non-origin contribution	60% of websites fetch content from more than 5 non-origins Non-origins make significant contributions to content—30% of objects and 35% of bytes in the median case Contribution of non-origins to page load time is low (80^{th} percentile is 35%) due to browser optimizations Images dominant object type served from origins, but Javascript accounts for sizeable fraction of non-origin objects Advertising and analytics services account for most non-origin objects, but CDNs account for most bytes

Table 2: Summary of key takeaways from our analysis with respect to various web page complexity metrics.

consider variation across samples when studying how complexity impacts variability in page load times.

4. CHARACTERIZING COMPLEXITY

Our analysis of our measurement dataset is two-pronged. First, in this section, we analyze web pages with respect to various complexity metrics. Next, in Section 5, we analyze the impact of these metrics on performance. Note that our focus is on capturing the complexity of web pages as visible to browsers on client devices; we do not intend to capture the complexity of server-side infrastructure of websites [43].

We consider two high-level notions of web page complexity. *Content complexity* metrics capture the number and size of objects fetched to load the web page and also the different MIME types (e.g., image, javascript, CSS, text) across which these objects are spread. Now, loading `www.foo.com` may require fetching content not only from other internal servers such as `images.foo.com` and `news.foo.com`, but also involve third-party services such as CDNs (e.g., Akamai), analytics providers (e.g., Google analytics), and social network plugins (e.g., Facebook). *Service complexity* metrics capture the number and contributions of the various servers and administrative origins involved in loading a web page.

We begin with the content-level metrics before moving on to service-level metrics. In each case, we present a breakdown of the metrics across different popularity rank ranges (e.g., top 1–1000 vs. 10000–20000) and across different categories of websites (e.g., *Shopping* vs. *News*). Here, we only show results for one of the vantage points as the results are (expectedly) similar across vantage points. Table 2 summarizes our key findings for the various complexity metrics.

4.1 Content complexity

Number of objects: We begin by looking, in Figure 2, at the total number of object requests required, i.e., number of HTTP GETs issued, to load a web page. Across all the rank ranges in Figure 2(a), loading the base web page requires more than 40 objects to be fetched in the median case. We also see that a non-trivial fraction (20%) of websites request more than 100–125 objects on their landing web page, across the rank ranges. While the top 1–400 sites load more objects, the distributions for the different rank ranges are qualitatively and quantitatively similar; even the lower rank websites have a large number of requests.

Next, we divide the sites by their Alexa categories. For clarity, we only focus on the top-two-level categories from Alexa. To ensure that our results are statistically meaningful, we consider only the categories that have at least 50 websites in our dataset. The breakdown across the categories in Figure 2(b) shows a pronounced difference between categories; the median number of objects requested on *News* sites is nearly 3× the median for *Business* sites. We suspect that this is an artifact of *News* sites tending to cram in more content on their landing pages compared to other sites to give readers quick snippets of information across different news topics.

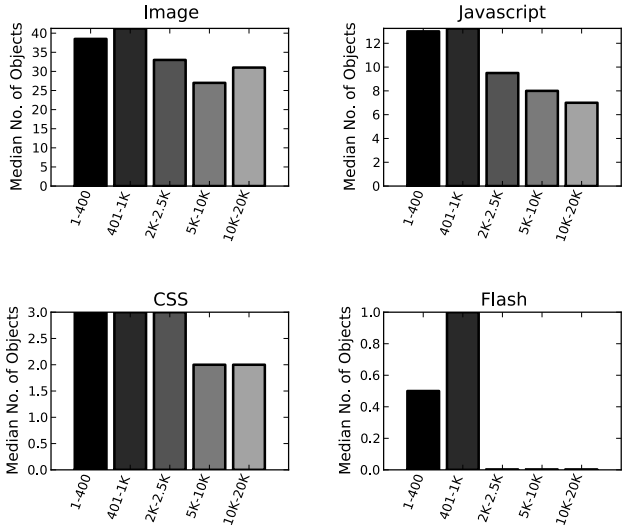


Figure 3: Median number of requests for objects of different MIME-types across different rank ranges.

Types of objects: Having considered the total number of object requests, we next consider their breakdown by content MIME types. For brevity, Figure 3 shows only the median number of requests for the four most popular content types across websites of different rank ranges. The first order observation again is that the different rank ranges are qualitatively similar in their distribution, with higher ranked websites having only slightly more objects of each type.

However, we find several interesting patterns in the prevalence of different types of content. While it should not come as a surprise that many websites use these different content types, the magnitude of these fractions is surprising. For example, we see that, across all rank ranges, more than 50% of sites fetch at least 6 Javascript ob-

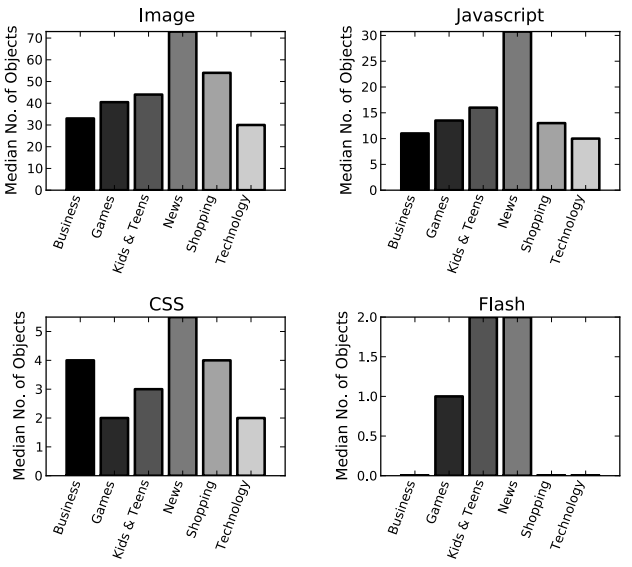


Figure 4: Median number of requests for objects of different MIME-types for different categories.

jects. Similarly, more than 50% of the sites have at least 2 CSS objects. The median value for Flash is small; many websites keep their landing pages simple and avoid rich Flash content. These results are roughly consistent with recent independent measurements [31].

Figure 4 shows the corresponding breakdown for the number of objects requested of various content types across different categories of websites. Again, we see the *News* category being dominant across different content types. As previously seen in Figure 2(b), *News* sites load a larger number of objects overall compared to other site categories. Hence, a natural follow-up question is whether *News* sites issue requests for a proportionately higher number of objects across all content types. Therefore, for each website, we normalize the number of objects of each content type by the total number of objects for that site. The distribution of the median values of the normalized fraction of objects of various content types (not shown) presents a slightly different picture than that seen with absolute counts. Most categories have a very similar normalized contribution from all content types in terms of the median value. The only significant difference we observe is in the case of Flash objects. Figure 5 shows that *Kids and Teens* sites have a significantly greater fraction of Flash objects than sites in other categories.

Bytes downloaded: The above results show the number of objects requested across different content types, but do not tell us the contribution of these content types to the total number of bytes downloaded. Again, for brevity, we summarize the full distribution with the median values for different website categories in Figure 6. Surprisingly, we find that Javascript objects contribute a sizeable fraction of the total number of bytes downloaded (the median fraction of bytes is over 25% across all categories). Less surprising is that images contribute a similar fraction as well. For websites in the *Kids and Teens* category, like in the case of number of objects, the contribution of Flash is significantly greater than in other categories. As in the case of the number of objects, we see no significant difference across different rank ranges.

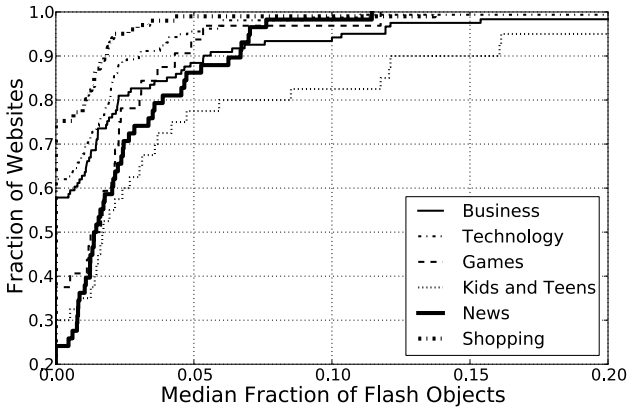


Figure 5: Fraction of objects accounted for by Flash objects, normalized per category.

4.2 Service complexity

Anecdotal evidence suggests that the seemingly simple task of loading a webpage today requires the client-side browser to connect to multiple servers distributed across several administrative domains. However, there is no systematic understanding of how many different services are involved and what they contribute to the overall task. To this end, we introduce several *service complexity* metrics.

Number of distinct servers: Figure 7 shows the distribution across websites of the number of distinct webservers that a client contacts to render the base web page of each website. We identify a server by its fully qualified domain name, e.g., `bar.foo.com`. Across all five rank ranges, close to 25–55% of the websites require a client to contact at least 10 distinct servers. Thus, even loading simple content like the base page of websites requires a client to open multiple HTTP/TCP connections to many distinct servers. Also, Figure 7(b) mirrors the earlier result from Figure 2(b); *News* sites have the most number of distinct servers as well.

Number of non-origin services: Not all the servers contacted in loading a web page may be under the web page provider's control. For example, a typical website today uses content distribution networks (e.g., Akamai, Limelight) to distribute static content, analytics services (e.g., google-analytics) to track user activity, and advertisement services (e.g., doubleclick) to monetize visits.

Identifying non-origins, however, is slightly tricky. The subtle issue at hand is that some providers use multiple origins to serve content. For example, `yahoo.com` also owns `yimg.com` and uses both domains to serve content. Even though their top-level domains are different, we do not want to count `yimg.com` as a non-origin for `yahoo.com` because they are owned by the same entity. To this end, we use the following heuristic. We start by using the two-level domain identifier to identify an origin; e.g., `x.foo.com` and `y.foo.com` are clustered to the same logical origin `foo.com`. Next, we consider all two-level domains involved in loading the base page of `www.foo.com`, and identify all potential non-origin domains (i.e., two-level domain not equal to `foo.com`). We then do an additional check and mark domains as belonging to different origins only if the authoritative name servers of the two domains do not match [33]. Because `yimg.com` and `yahoo.com` share the same authoritative name servers, we avoid classifying `yimg.com` as having a different origin from `yahoo.com`.

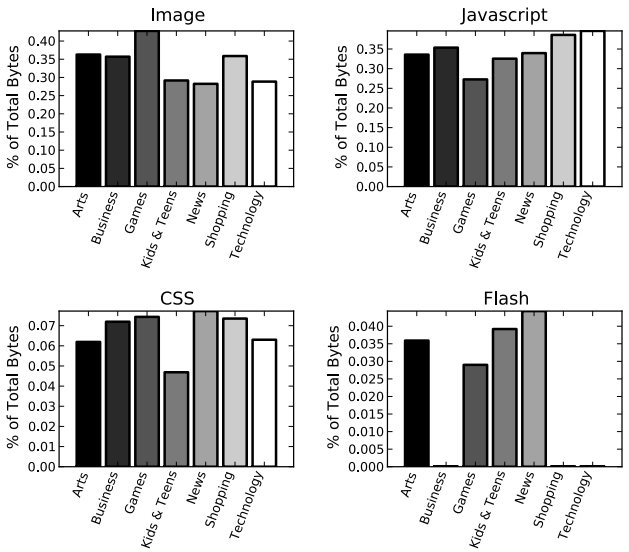


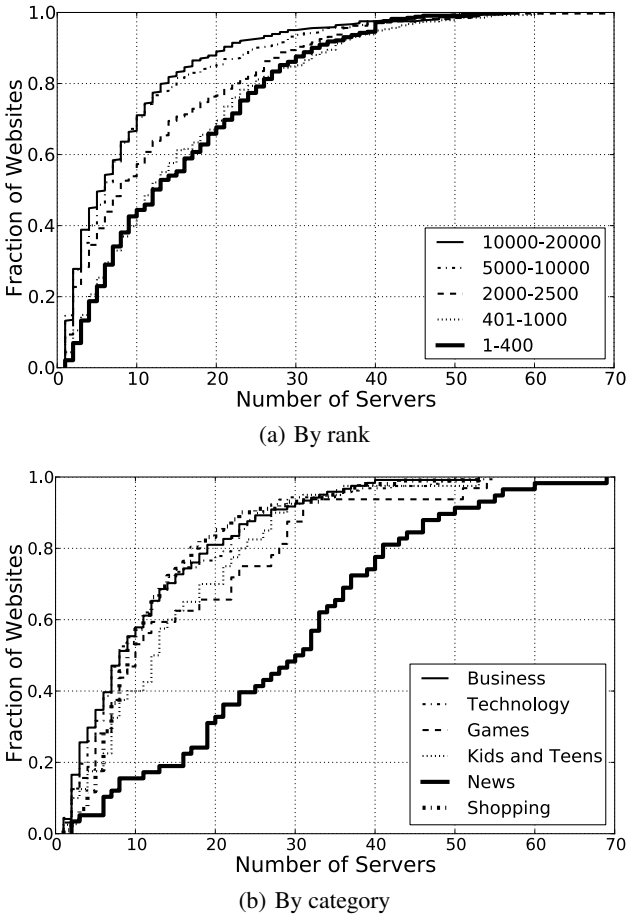
Figure 6: Median normalized contribution of different MIME types to total bytes downloaded.

Figure 8 shows that across the different rank ranges and categories, clients need to contact servers in at least 10 different origins for 20–40% of websites. The presence of non-origin content is even more pronounced on *News* sites; more than 40% of *News* sites serve content from over 20 non-origin providers. On further inspection, we find that because the landing pages of *News* sites have to provide content that spans multiple user interests (e.g., sports, weather) they provide links to non-origin affiliates that serve such content as well.

Contribution of non-origin services: The previous result simply counts the number of distinct domains contacted. Next, we quantify the *contribution* of the non-origin domains along three dimensions: fraction of objects, fraction of bytes, and fractional contribution to total page load time.

Figure 9 shows that, in the median case, over 30% of the total number of objects and over 35% of the total number of bytes downloaded are from non-origin services. At the same time, we see that the distribution is pretty heavy-tailed; for 20% of websites, non-origin services account for roughly 80% of the objects and total bytes.

The total number of objects or bytes may, however, not directly translate into download time because modern browsers can parallelize requests to multiple servers. Now, parallelization also makes it inherently difficult to exactly determine the time contribution of non-origins. In light of this, we use three alternative ways to measure the non-origin contribution to total page load time: (1) the "wall-clock" time where the browser is retrieving content from at least one non-origin (labeled "Time: At Least 1 Non-Origin"), (2) the ratio of the sum of all time spent in downloading non-origin content to the total time spent downloading all content (labeled "Time: Total Cycle Contribution"), and (3) emulating the act of loading the page by disabling all non-origin content using custom Adblock filters (labeled "Time: Block Non-Origin"). We see in Figure 9 that in the median case, content from non-origins contributes to only 15% of the page load time in terms of the *At Least 1 Non-Origin* and around 25% for the *Total Cycle Contribution*. These results suggest that though non-origin services play a significant part of the web ecosystem in terms of the fraction of content they contribute, browser optimizations (e.g., pipelining and parallelizing requests to distinct servers) lower their impact on page load times.



(a) By rank

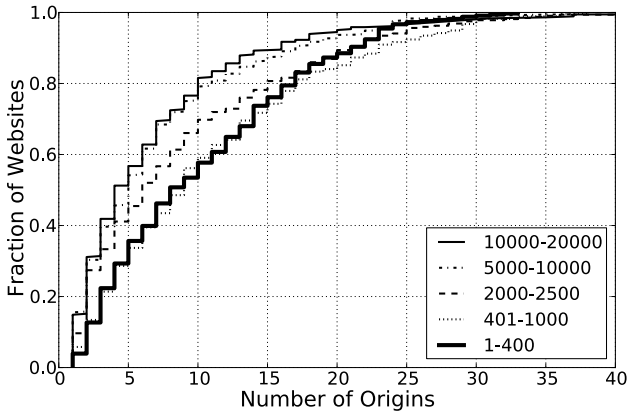
(b) By category

Figure 7: Number of distinct servers contacted to load the base webpage for websites across different rank ranges and categories.

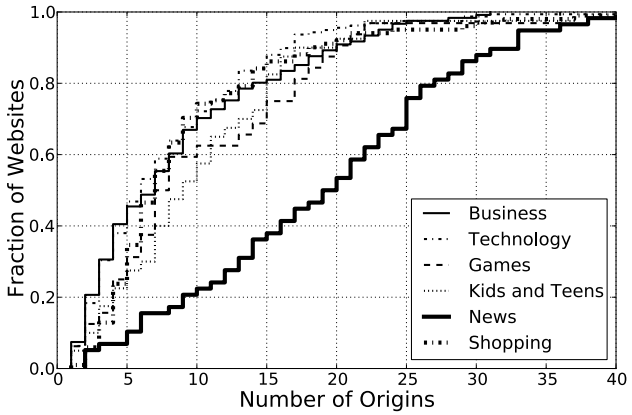
4.3 What do non-origins offer?

A natural question is what types of content and services do non-origins provide. Beyond a basic curiosity of what non-origin content includes, this also has important performance implications. For example, if most non-origin objects constitute content essential for the user experience, then it might be difficult for website providers to directly optimize their delivery or client-side blocking of non-origin content would adversely affect user experience. But, if most non-origin objects are from services for displaying ads or tracking users, they could potentially be consolidated or optimized. Therefore, in this section, we do a more in-depth analysis of the MIME-types of objects served by non-origins, how they differ from objects served from origins, and also identify the class of services that these non-origins provide.

Content type breakdown: As a first step, we want to understand what *types of content* are served by non-origins across different websites. Figure 10 shows the breakdown of the different content types served by non-origins, both in terms of the number of objects and their size in bytes. This breakdown is shown for the median



(a) By rank



(b) By category

Figure 8: Number of distinct origins needed to load the base web page for websites across different rank ranges and categories.

website, i.e., the website that loads the median number of objects (or median number of bytes) for each content type. Interestingly, we find that while the vast fraction of the number of objects served by non-origins are images, the relative fraction in terms of number of bytes served is much lower. This is at odds with the normal expectation that the contribution of images to bytes fetched will be larger than their contribution to the number of objects, since images are typically larger than Javascript and CSS objects. Investigating this further, we find that this discrepancy is an artifact of the use of small *gifs* fetched from non-origins for analytics services [14]. We illustrate this point in Figure 11, which shows the distribution of the number of objects and object size for each MIME-type. We see that though images are the most common type of content, the median size of an image is less than 2 KB—more than an order of magnitude smaller than the median size of a Flash object.

Origin vs. non-origin content: Next, we proceed to analyze if the content served by non-origins differs significantly from that served by the origin sites themselves. Figure 12 shows the contribution of different MIME-types to the number of objects fetched from origins and non-origins on the median website. The most noticeable difference is that non-origins serve a much higher fraction of Javascript objects while origins serve a greater fraction of images than non-origins.

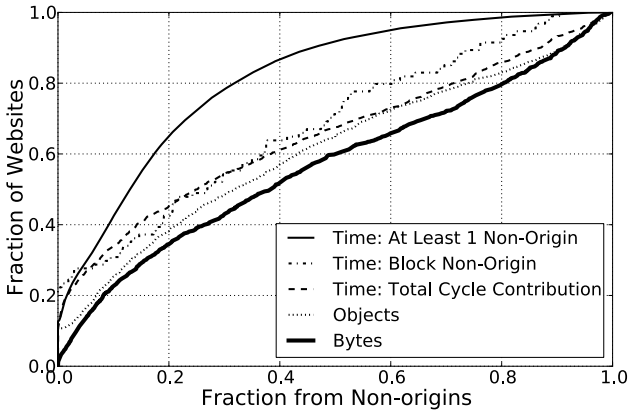


Figure 9: Contribution of non-origin services with respect to number of objects, number of bytes, and download time.

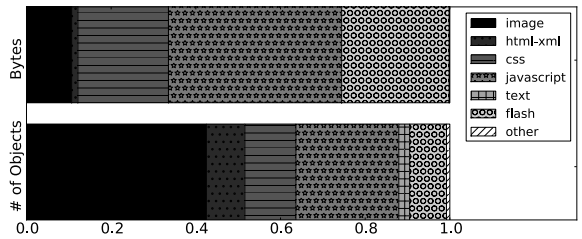
Classifying non-origins: Beyond types of content, we analyze the types of services offered by non-origins. For this study, we rank non-origin domains based on the number of websites in which they occur. Then, for the 200 most popular non-origins, we identify the services they offer by combining three sources of information. First, we lookup each non-origin domain in Alexa's categorization of web domains. Second, we use information from CrunchBase[4] to identify the type of company that runs the non-origin service. Last, we manually classify the remaining non-origins based on information gleaned from their website or keyword-based heuristics on the objects being fetched from them.

Table 3 presents a breakdown of the types of services these top 200 non-origins offer and the number of origins in which each category appears. Here, we only consider the 669 origins, in which all the non-origin objects belong to one of these top 200 non-origins. Unsurprisingly, the top two categories of non-origin services are *Analytics* (e.g., google-analytics and quantserve) and *Advertising* (e.g., doubleclick and googleadservices). However, even beyond these two service types, we see that each of the non-origin service types are seen on a significant fraction of the 669 origins.

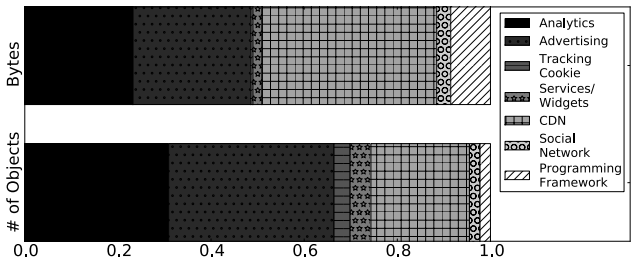
To give an example of the types of non-origin services we encounter, Table 4 shows the top 10 non-origins with the type of service they provide and the fraction of sites on which they appear. While many of these are very recognizable ad and analytics providers, we were surprised by some of the less recognized names appearing in a significant fraction of websites. For example, among the top 20, we found other lesser known services like bluekai.com, invitemedia.com, and imrworldwide.com, that each appeared in more than 5% of the websites (not shown).

Finally, we examine the contribution of these non-origin service types on a typical web page. For this analysis, we only consider the 669 websites where all non-origins on their base web page belong to the top 200 non-origins. On each of these 669 websites, we compute the fraction of the number of objects and bytes fetched from non-origins that are attributable to each service type. Figure 10(b) plots these fractions for the median website. In keeping with the relative popularity of non-origin services of different types in the top 200 non-origins, Analytics and Advertising account for most non-origin objects. However, content fetched from CDNs dominate with respect to the number of bytes.

[4] `http://www.crunchbase.com`



(a) By MIME-type

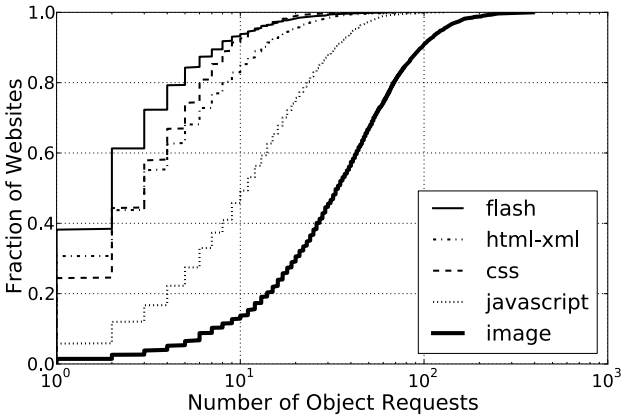


(b) By service type

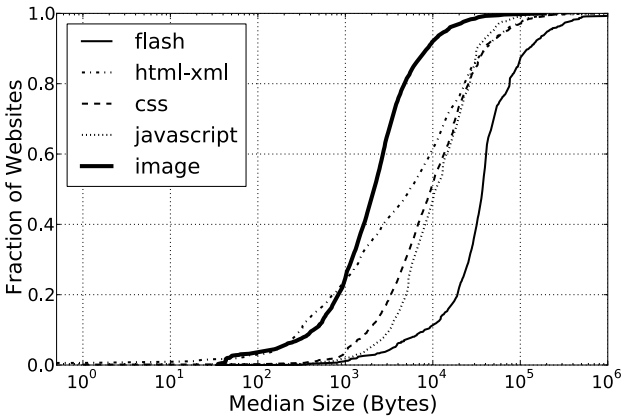
Figure 10: Normalized contribution of objects from non-origin services in the median case.

Type of service	Number	Found in no. of origins
Analytics	65	593
Advertising	64	233
Tracking Cookies	23	137
Services/Widgets	21	142
CDN	18	218
Social Networking	5	98
Programming API	4	96
Total	**200**	**669**

Table 3: Breakdown of the types of services provided by top 200 non-origins.



(a) No. of objects



(b) Object size

Figure 11: Distributions across content types of the number of objects and median object size from non-origins.

4.4 Summary of main observations

In summary, our main observations in this section are as follows:

- A website's rank is not a significant indicator of the content complexity, at least within the top 20K websites.
- However, a website's category does matter; *News* sites load significantly more content than others from a lot more servers and origins, while *Kids and Teens* sites have significantly more Flash content than others.
- Most websites load a surprisingly large number of CSS and Javascript objects.
- Content from non-origins represents a significant fraction of objects and bytes on most web pages, but their impact on download time is relatively low.

5. IMPACT ON CLIENT PERFORMANCE

In the previous section, we measured a range of content-complexity and service-complexity metrics. In this section, we tackle the natural follow-up question: which of these metrics have the most impact on *performance*.

We consider two performance measures to characterize page load times. RenderStart measures the time at which the browser has completed parsing the HTML and has fetched sufficient content to begin rendering the page, while RenderEnd measures the total time to fetch and render all content on the web page. For each measure, we are interested in both the *typical* load time for each website and the *variability* in load time across different samples.

To put our analysis in perspective, Figure 13 shows the distribution of the median and 90^{th} percentile of the RenderEnd values for each site across several measurements from one of our vantage points. Rather surprisingly, more than 50% of sites have a median RenderEnd higher than 2 seconds. (We also validated these seemingly high page load times from independent measurements from the HTTP Archive project [3].) User studies and industry surveys show that users are likely to be frustrated beyond this two second threshold [28]. Thus, it is critical to systematically understand what are the key factors affecting page load times.

We use correlation and regression based analysis to identify the key complexity metrics that are the best indicators of page load times and the variability in them. In this analysis, we use a range of complexity metrics from the previous section—the {absolute value or fraction} of {objects, bytes, servers, origins, non-origins} characterized by {content MIME-type, service type}, and whether loaded from {origin, non-origin, either}. We also use other aggre-

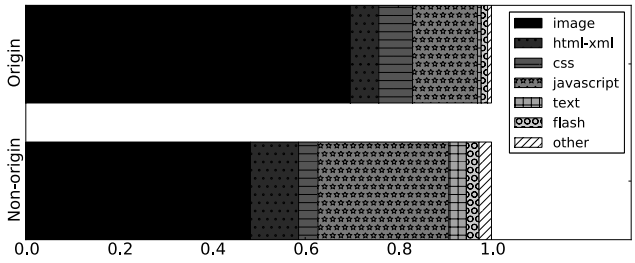


Figure 12: Comparison of types of content served from origin and non-origin servers on the median website.

Rank	Name	Fraction of sites	Type
1	google-analytics.com	0.58	Analytics
2	doubleclick.net	0.45	Ads
3	quantserve.com	0.30	Analytics
4	scorecardresearch.com	0.27	Analytics
5	2mdn.net	0.24	Ads
6	googleadservices.com	0.18	Ads
7	facebook.com	0.17	Social net
8	yieldmanager.com	0.16	Ads
9	atdmt.com	0.14	Analytics
10	googleapis.com	0.12	Prog. API

Table 4: Classification of the services provided by the top-10 non-origin service providers.

gate metrics such as the size of the maximum object fetched. For brevity, we only present results for metrics that turned out to be the most dominant indicators of either absolute load times or their variability.

5.1 Load times

Correlation: First, we analyze the correlation between RenderEnd (RenderStart) and various complexity metrics. For this analysis, we compute for each website the median values of RenderEnd (RenderStart) across multiple measurements of that website and the median value of various complexity metrics. Then, across all websites in our dataset, we compute the Pearson and Spearman rank correlation coefficients between the two load time measures and various complexity metrics. Since the results are similar for RenderStart and RenderEnd, we present only the results for RenderEnd. Also, the results are similar for both Pearson and Spearman correlations; hence, for brevity, we only show the Spearman values. To ensure that the correlations we observe are not artifacts of a particular measurement site, we consider each of the 4 vantage points separately. Figure 14 shows the Spearman correlation coefficients with respect to various complexity metrics in decreasing order of the median value across the different measurement sites. Across all 4 measurement sites, we see that the five most correlated metrics are the total number of objects loaded, the number of these objects that are Javascripts, the total webpage size, the number of servers, and the number of origins contacted in loading the page.

Figure 15 further visually confirms the strong correlation between RenderEnd and the number of objects requested. Here, we bin websites based on the number of objects on their base web page. Then, for each bin, we construct a box-and-whiskers plot showing the median, 25^{th} percentile, and 75^{th} percentile plot in the "box" and the min/max values for the whiskers. Further, tying

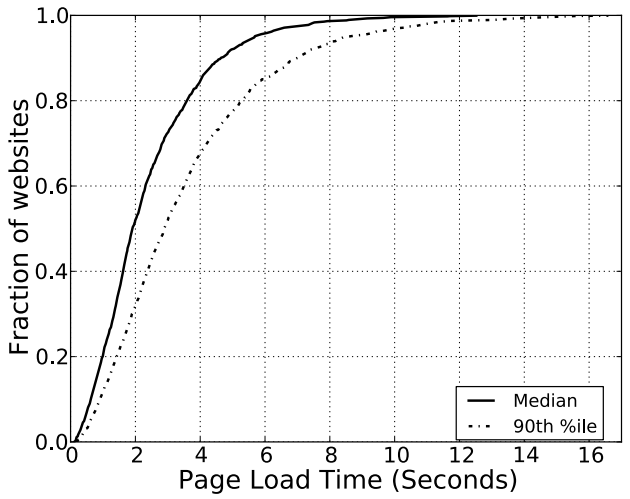


Figure 13: Distribution of RenderEnd times across all websites.

the fact that number of object requests is the most dominant indicator of load times with our observation from Section 4 that *News* sites fetch a significantly larger number of objects than other sites, Figure 16 shows that the page load times for *News* sites are indeed much higher than for other sites.

Regression: The correlation analysis tells us which metrics are good indicators of page load times. Next, we attempt to identify a minimal set of metrics to estimate load times. For this, we augment the above correlation analysis by building a linear regression model using the LASSO technique [17]. Each *feature* in this model represents one of the various complexity metrics presented in the previous section. We use LASSO instead of simple linear regression because it produces a *sparser* model; thus, models with LASSO are more robust. To further avoid overfitting, we use a $k \times 2$ cross validation technique. Here, in each run, we partition the set of websites into two halves—a training set and a testing set. For each run, we run the LASSO procedure and record the coefficients for each feature. Then, we build an aggregate model using the average values of the individual coefficients across all runs.

Figure 17 shows the normalized root mean-squared error (NRMSE),[5] as a function of the top k selected features. In this figure, we sort the features based on the magnitude of their weights in the model after the cross-validation procedure described above.[6] Then, we emulate the effects of using a model that uses only the top k features. As a point of comparison, we also considered a naive estimator that simply predicts the mean value of RenderStart and RenderEnd; its NRMSE was around 50% worse than the LASSO estimate (not shown).

We see two key results here. First, the set of top k features roughly mirrors the correlation result we saw earlier. One notable exception is that the total size is relegated down the list. We spec-

[5]If $\hat{X}$ is a vector representing estimates of page load times, and X contains the true values, then NRMSE = $\frac{\sqrt{E[(\hat{X}-X)^2]}}{\max(X)-\min(X)}$

[6]One obvious concern is whether the magnitude of the weights are meaningful if the different features and the load time are in different "scales". A pre-processing step in LASSO re-scales all features to have zero mean/unit variance and also normalizes the load metric to have zero mean. Thus, the magnitude measures the relative importance of the metric and avoids these scale-related concerns.

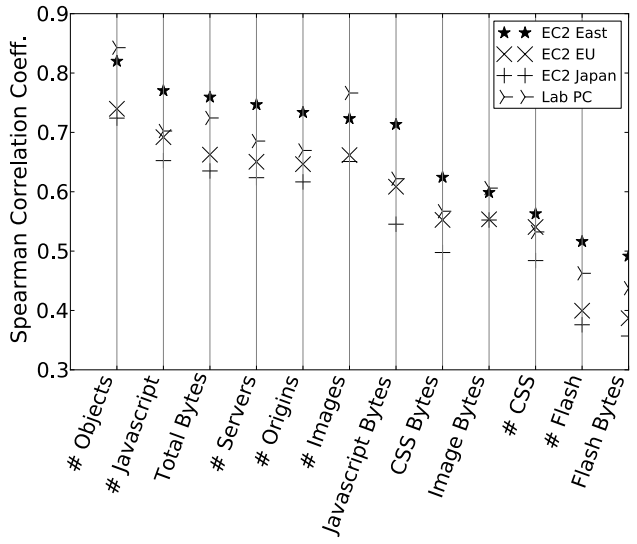


Figure 14: Correlation between RenderEnd and various complexity metrics.

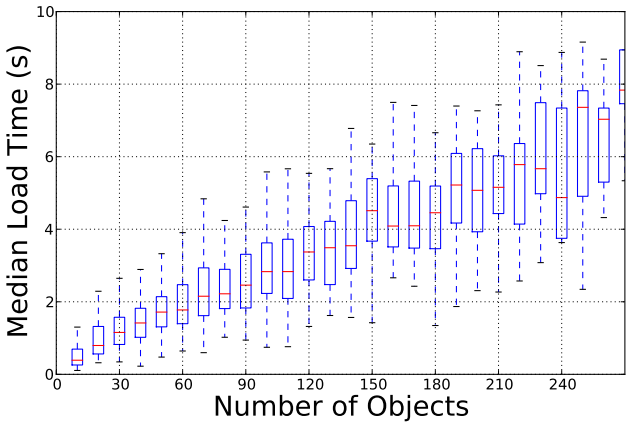


Figure 15: Box-and-whiskers plot confirming the correlation between RenderEnd and number of objects.

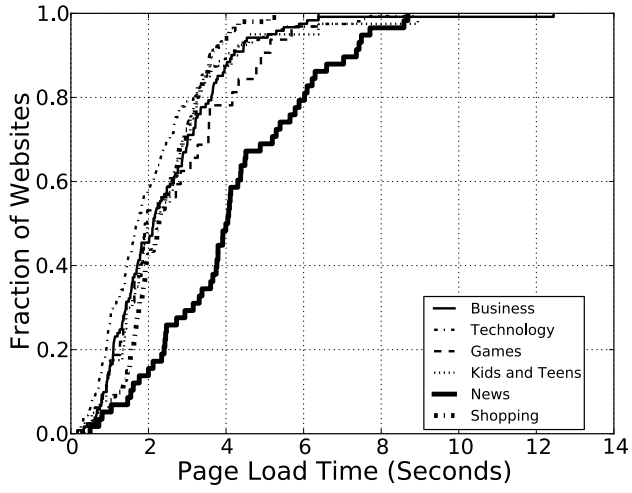


Figure 16: Page load times for websites of different categories.

ulate that because the base web pages of websites are typically not that large, page load times are dominated by the number of RTTs rather than the number of bytes transferred. Thus, having chosen the number of requests and number of scripts as the top 2 features, the marginal contribution of total size is low. Second, we find that the prediction error flattens after the first 4-5 features are selected. In other words, we see a natural diminishing returns effect of adding other features. Identifying a small, but predictive set of features is a critical result for both website designers and performance optimization tools.

Figure 18 visually confirms the goodness-of-fit. The X-axis represents the actual page load time (RenderEnd), and the Y-axis represents the load time predicted by the regression model. One interesting artifact is a *horizontal* cluster near the origin, where the model predicts the same value of RenderEnd for around 30 sites. On further investigation, we discovered that these websites have very sparse landing pages that have little to no content. Thus, the values of the top 5 metrics were zero and the prediction model outputs a constant value.

Having determined the top k complexity metrics for the entire population of websites in our study, we next analyze if there is a significant difference across website categories. We repeat the regression analysis for each category separately. Table 5 shows the top 2 metrics identified by the regression for each category. It also shows the cardinality of the set intersection between the top 5 metrics for each category and the top 5 metrics from the aggregate regression model across all websites (from Figure 17).

First, we see that the top 2 metrics are quite diverse across the different categories. For example, the number of images and requests are most critical for `News` sites, but the total size of Javascript objects and the number of servers are most critical for `Games` sites. Second, there is a significant overlap in the top 5 metrics between the per-category models and the overall model. In fact, for most categories, the metrics in the intersection are the number of objects, the number of servers, and the number of Javascript objects, which are the top 3 metrics in Figure 17. This suggests that while there is some value in customizing the prediction model to identify the critical metrics, the aggregate model is a very useful starting point in itself.

We also experimented with an enhanced regression model where we added nonlinear terms, e.g., $\log(X)$ and X^2, for each feature X. Adding these nonlinear terms does not improve the prediction error and does not change the stability of the top k feature set. We do not present these results given space limitations.

5.2 Variability in load times

So far, we studied the impact of various complexity metrics on median page load times. Next, we analyze if the same set of factors impact how load times *vary*, or if a different set of metrics are critical for predicting variability. Here, we restrict our analysis to only focus on websites where the complexity metrics such as page size and number of object requests do not change significantly across measurements. That is, the variability in RenderEnd cannot be attributed simply to a change in the content of the page across our measurements.

As a starting point, we measure variability in load times for any particular website as the difference between the 75^{th} and 25^{th} per-

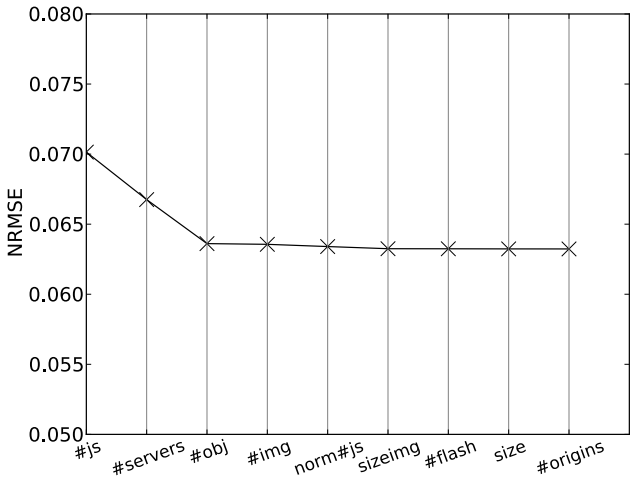


Figure 17: Normalized mean-squared error as a function of top k coefficients.

Category	Top 2	Top 5 Intersection
Business	# objects, # images	3
Technology	# js, # origins	3
Games	# servers, size js	3
Kids and Teens	# js, # objects	3
News	# images, # objects	3
Shopping	size css, # js	3

Table 5: Highest-impact complexity metrics in the regression model for different website categories, and their intersection with aggregate regression model.

centile values of RenderEnd across different measurements for that website; we consider the difference between the 75^{th} and 25^{th} percentiles instead of between the maximum and minimum values to discount for any client-side effects. Then, we correlate the absolute and normalized (by the median) value of this variability versus the various complexity metrics. As seen in Figure 19, in comparison to the earlier correlation result, we find two differences. First, the correlations are weaker in general (e.g., the highest value is 0.65). Second, the number of servers is the most dominant factor in this case, instead of the number of objects, which was the dominant indicator of absolute load times.

5.3 Summary of main observations

The key takeaways from our analysis of load times are:

- The top five complexity metrics that determine RenderStart and RenderEnd are the total number of objects loaded, the number of these objects that are Javascripts, the total webpage size, and the number of servers and origins contacted in loading objects on the page.
- We can build a sparse model for predicting page load times with a normalized mean squared error less than 0.1.
- Variability of load times is less correlated with our complexity metrics, but number of servers is its most dominant indicator rather than the number of objects.

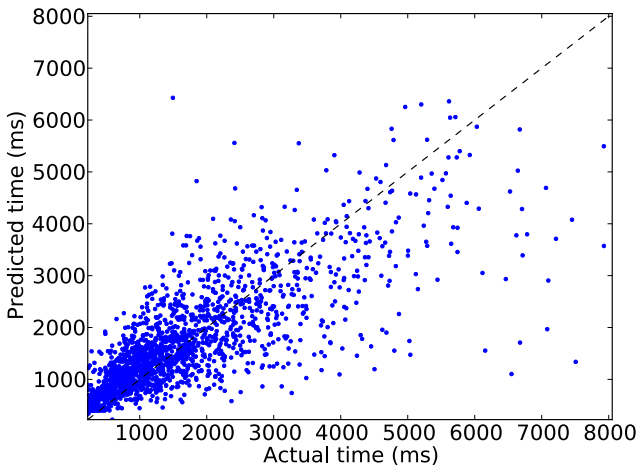


Figure 18: Scatter plot of page load times predicted by the regression model that uses the top 5 features versus the actual page load times.

6. DISCUSSION

The previous sections provide a good first-order understanding of how website complexity affects the user experience. We acknowledge that there are likely to be a much wider range of factors that can affect a user's web experience: "deeper" non-landing pages, diversity in client-side platforms, use of client and provider tools for customizing websites, and other forms of personalization (e.g., services that require a login). In this section, we present a preliminary discussion of such factors.

Landing vs. non-landing pages: Our study focused on the landing pages of websites. As a preliminary study to evaluate how "deeper" non-landing pages might differ, we consider a random sample of 100 sites. For each such site, we follow other links to pages within the same domain (e.g., `www.foo.com` has a link to `www.foo.com/bar` or `x.foo.com/bar`) and compute the various complexity metrics for each such non-landing page. For each site and metric, we then look at the difference between the base site and the median across these landing pages, and normalize this difference by the value for the base site. Figure 20 shows the distribution across sites of these normalized differences for five key metrics: number of requests, number of servers, fraction of non-origin objects, page size, and download time. We see that other than from the perspective of the fraction of non-origin objects metric, websites on which non-landing pages are less complex than the base site far outweigh the sites on which the opposite is true. We do, however, see a long negative tail with the non-landing pages of some sites being up to 2× more complex than the base site.

Choice of browser: The choice of browser does not affect our complexity analysis in Section 4. However, browsers may vary in the specific strategies in how they parallelize requests, optimize scripts and so on; this could affect the load time analysis in Section 5. One additional concern when comparing load time results across browsers is that the semantics of `onLoad` might vary [7]. (This is not a concern for our paper because all our measurements use the same version of Firefox.) We leave for future work the task of understanding these effects.

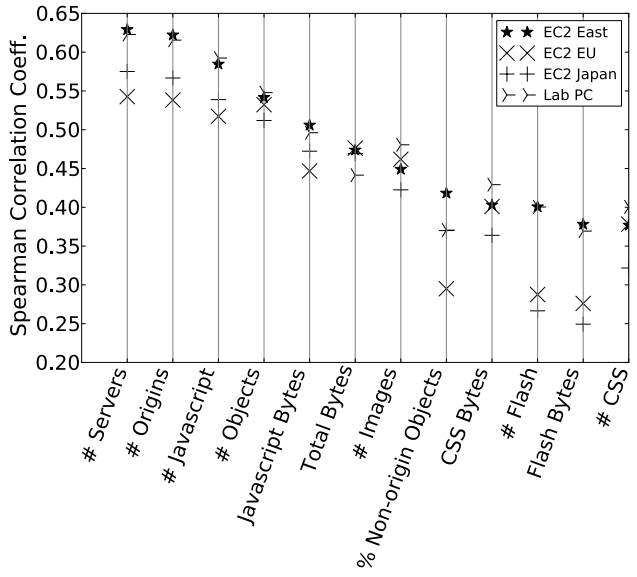


Figure 19: Correlation coefficients between the different metrics and the variability in RenderEnd.

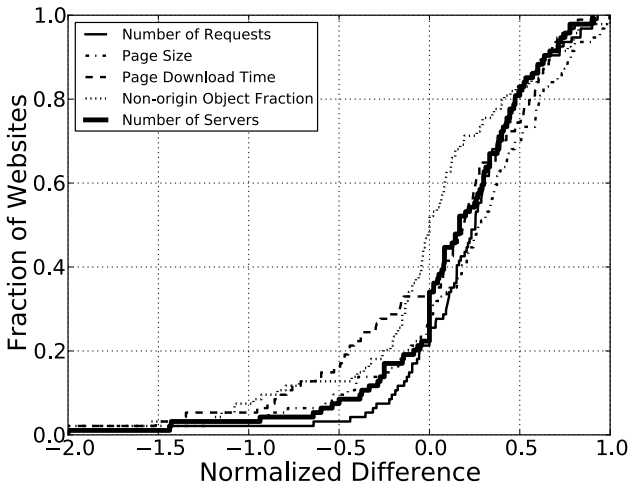


Figure 20: Preliminary result showing that non-landing pages are less complex than landing pages for most sites.

Personalized web services: Some services present different landing pages to users who have subscribed or logged in. For example, `facebook.com` and `igoogle.com` have content that is personalized to the specific user logged in. The key to understanding these effects is to emulate user profiles that are representative of a broad spectrum of browsing habits (e.g., casual vs. expert users). While this is relevant to our broader theme, it is outside the scope of this paper.

Interaction with client-side plugins: Users today deploy a wide range of client-side browser extensions to customize their own web browsing experience. For example, two popular Firefox extensions block advertisements (`Adblock`) and scripts (`NoScript`). We conducted a preliminary study spanning 120 randomly chosen websites from our overall list on how these extensions impact the complexity metrics. We avoid page load time here because these extensions alter the user experience (i.e., it is not showing the same content) and it is unfair to compare the time in this case.

Figure 21 compares two separate browser instances—one with the extension enabled, and the other with the browser instance from our previous measurements. Here, we install each extension with its default configuration. We only show the complexity metrics that were dominant indicators of page load time and variability in load time: number of objects and number of servers. The median number of objects requested reduces from 60 on the vanilla browser to 45 with Adblock and 35 with NoScript. The reduction in the number of servers is even more marked—the median value drops from 8 to 4 with Adblock and to 3 with NoScript. `NoScript`'s effect is more pronounced than that of `Adblock` because disabling scripts can in turn filter out objects that would have been fetched as a consequence of the script's execution.

It is unclear though if the reduction in complexity caused by these client-side controls is entirely good, even though it likely improves page load times. We do not know how this affects the user experience that the website provider intended for the user (e.g., is some useful content being blocked?) and how these may affect the provider's business interests (e.g., ad click/conversion rates).

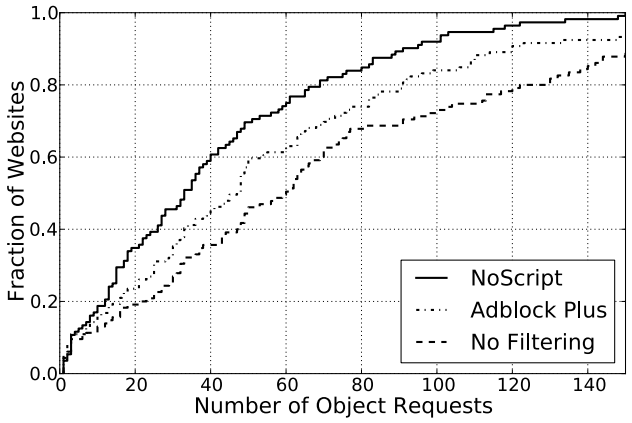
Customization for client platform: In parallel to the evolution of web pages, the diversity of client platforms (e.g., mobile phones, tablet computers, and even televisions) used for web access has also increased. These platforms vary in their connectivity, display attributes, and user interface. Consequently, providers are interested in customizing the web experience for these platforms.

We considered 120 randomly chosen websites that offer customized web pages for mobile phones. We visited these sites once with the default browser setting and once with the browser instrumented to emulate an iPhone (by spoofing the UserAgent string). Figure 22 shows that the phone-specific customization dramatically reduces the number of objects fetched and the number of servers contacted. Again, we focus only on complexity metrics and not the page load time because it does not represent a reasonable comparison (e.g., time on actual mobile phones would be different than on a spoofing browser). However, as in the case of client-side controls, it is not clear if this customization affects the user experience compared to a desktop-based experience (e.g., was some content dropped).

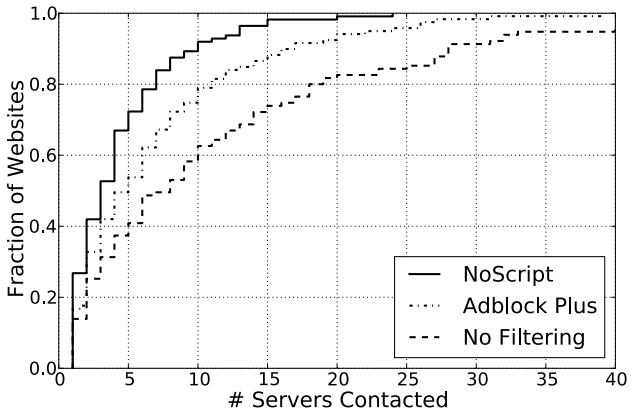
Optimizations for improving performance: Many tools like PageSpeed from Google [2] suggest optimizations such as compressing images, combining requests for small images and CSS files, and "minify-ing" Javascript objects to improve website performance. Based on some sample websites, we found that *minify js* appears as a "high priority" suggestion for several websites.[7]

To analyze how this optimization would help websites in the wild, we emulate the effect of running it on each website in our dataset. Figure 23 shows the savings this optimization could provide in terms of the fraction of total bytes of Javascript and the fraction of the total size of the web page downloaded. We see that for the vast majority of websites, the potential savings is quite small. While this result is preliminary and does not explore all possible optimizations, it does hint that optimizations that are broadly perceived as high-priority may not yield high gains for all websites. Thus, we need to explore more systematic tools and new avenues to improve page load times.

[7]This is a code compacting tool that removes unnecessary white spaces, and comment lines to reduce the size of Javascript objects.

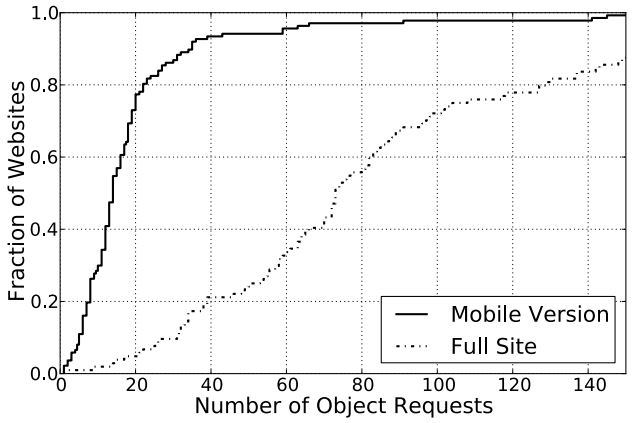


(a) No. of objects

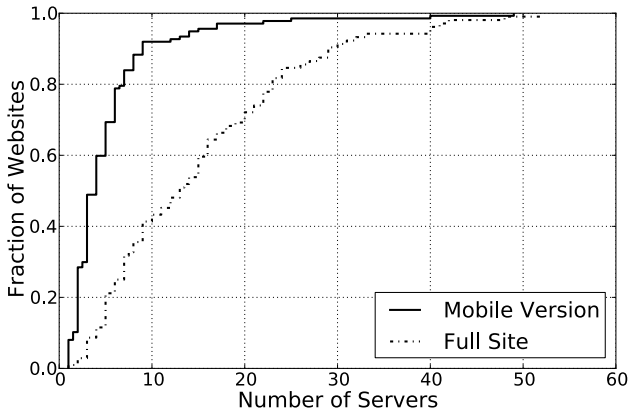


(b) Total no. of servers contacted

Figure 21: Reduction in number of objects and servers contacted with client-side filtering.



(a) No. of objects



(b) No. of servers

Figure 22: Reduction in number of objects and servers with phone-specific customization.

7. CONCLUSIONS AND FUTURE WORK

The increasing complexity of web pages and its impact on performance has been anecdotally well-recognized, but there have been no rigorous studies of the same. In this paper, we presented a first attempt at characterizing web page complexity and quantifying its implications. We characterized the complexity of web pages both based on the content they include and the services they offer. We find that a website's popularity is a poor indicator of its complexity, whereas its category does matter. For example, *News* sites load significantly more objects from many more servers and origins than other categories. Also, we found that though a significant fraction of objects and bytes are fetched from non-origins on most websites, the contribution of non-origins to page load time is minimal in comparison. Our correlation- and regression-based analysis showed that number of objects and number of servers are the dominant indicators of page load time and variability in page load times, respectively.

Our preliminary results also show that reducing this complexity and improving client performance is not straightforward. Though client-side filtering of particular content types may reduce page load times, it can adversely impact user experience and compromise revenue for web providers. On the other hand, website providers need to understand the impact that non-origin content on their web pages has on their users in order to customize for different client platforms.

As future work, we are continuing our efforts in several directions—deeper analyses of non-landing pages, studying the dependency between the various objects on a page, finding better indicators of performance and variability by focusing on websites within certain rank ranges and certain categories, and designing strategies to systematically balance the tradeoff between performance, user experience, and the provider's business interests.

8. REFERENCES

[1] Firebug. `http://getfirebug.com/`.

[2] Google Page Speed. `http://code.google.com/speed/page-speed/`.

[3] HTTP archive beta. `http://httparchive.org/`.

[4] HTTP archive specification. `http://groups.google.com/group/http-archive-specification/web/har-1-1-s%pec?hl=en`.

[5] Keynote systems. `http://www.keynote.com`.

[6] Let's make the web faster. `http://code.google.com/speed/articles/web-metrics.html`.

[7] Measuring the mobile web is hard. `http://matt-welsh.blogspot.com/2011/08/measuring-mobile-web-is-hard.htm%l`.

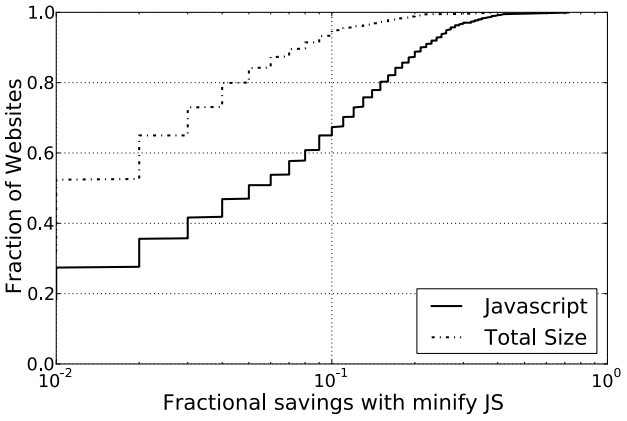


Figure 23: Bytes saved using *minify js*, one optimization from Google's Page Speed for reducing page load times.

[8] Mobify. `http://mobify.me`.

[9] The need for speed. `http://www.technologyreview.com/files/54902/GoogleSpeed_charts.pdf`.

[10] Opera Mini & Opera Mobile browsers. `http://www.opera.com/mobile/`.

[11] SPDY: An experimental protocol for a faster web. `http://www.chromium.org/spdy/spdy-whitepaper`.

[12] Spec web benchmarks. `http://www.spec.org/web2005/`.

[13] Strangeloop: Speed up your website. `http://www.strangeloopnetworks.com`.

[14] Web bug. `http://en.wikipedia.org/wiki/Web_bug`.

[15] When seconds count. `http://www.gomez.com/wp-content/downloads/GomezWebSpeedSurvey.pdf`.

[16] J. M. Kleinberg, S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. The web as a graph: Measurements, models and methods. In *Proc. COCOON*, 1999.

[17] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of Royal Statistical Society of Britain*, 1996.

[18] A. Bouch, A. Kuchinsky, and N. Bhatti. Quality is in the Eye of the Beholder: Meeting Users' Requirements for Internet Quality of Service. In *Proc. CHI*, 2000.

[19] A. Broder et al. Graph structure in the web. *Computer Networks*, 33(1), June 2000.

[20] B. Ager, W. MÃijhlbauer, G. Smaragdakis, and S. Uhlig. Web content cartography. In *Proc. IMC*, 2011.

[21] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, M. Stemm, and R. H. Katz. TCP behavior of a busy Internet server: Analysis and improvements. In *Proc. IEEE Infocom*, 1998.

[22] T. Benson, A. Akella, and D. Maltz. Unraveling the Complexity of Network Management. In *Proc. NSDI*, 2009.

[23] G. Candea. Toward Quantifying System Manageability. In *Proc. HotDep*, 2008.

[24] J. Cao, W. S. Cleveland, Y. Gao, K. Jeffay, F. D. Smith, and M. C. Weigle. Stochastic Models for Generating Synthetic HTTP Source Traffic. In *Proc. INFOCOM*, 2004.

[25] B.-G. Chun, S. Ratnasamy, and E. Kohler. NetComplex: A Complexity Metric for Networked System Designs. In *Proc. NSDI*, 2008.

[26] D. Galletta, R. Henry, S. McCoy, and P. Polak. Web Site Delays: How Tolerant are Users? *Journal of the Association for Information Systems*, 2004.

[27] D. Mosberger and T. Jin. httperf: A Tool for Measuring Web Server Performance. *SIGMETRICS Performance Evaluation Review*, 26(3), 1998.

[28] F. Nah. A study on tolerable waiting time: How long are Web users willing to wait? *Behaviour & Information Technology*, 23(3), May 2004.

[29] D. Fetterly, M. Manasse, M. Najork, and J. Wiener. A large-scale study of the evolution of web pages. In *Proc. WWW*, 2003.

[30] P. Gill, M. Arlitt, N. Carlsson, A. Mahanti, and C. Williamson. Characterizing Organizational Use of Web-based Services: Methodology, Challenges, Observations, and Insights. *ACM TWEB*, 2011.

[31] S. Ihm and V. S. Pai. Towards understanding modern web traffic. In *Proc. IMC*, 2011.

[32] E. Kiciman and B. Livshits. AjaxScope: A Platform for Remotely Monitoring the Client-Side Behavior of Web 2.0 Applications. In *Proc. SOSP*, 2007.

[33] B. Krishnamurthy and C. E. Willis. Privacy diffusion on the web: A longitudinal perspective. In *Proc. WWW*, 2009.

[34] B. Krishnamurthy, C. E. Willis, and Y. Zhang. On the use and performance of content distribution networks. In *Proc. IMW*, 2001.

[35] M. Lee, R. R. Kompella, and S. Singh. Active measurement system for high-fidelity characterization of modern cloud applications. In *Proc. USENIX Conference on Web Applications*, 2010.

[36] R. Levering and M. Cutler. The portrait of a common HTML web page. In *Proc. ACM Symposium on Document Engineering*, 2006.

[37] L. Meyerovich and R. Bodik. Fast and parallel web page layout. In *Proc. WWW*, 2010.

[38] J. C. Mogul. The case for persistent-connection HTTP. In *Proc. SIGCOMM*, 1995.

[39] A. Nazir, S. Raza, D. Gupta, C.-N. Chuah, and B. Krishnamurthy. Network level footprints of Facebook applications. In *Proc. IMC*, 2009.

[40] S. Gribble et al. The Ninja architecture for robust Internet-scale systems and services. *Computer Networks*, 35(4), Mar. 2001.

[41] F. Schneider, S. Agarwal, T. Alpcan, and A. Feldmann. The new Web: Characterizing AJAX traffic. In *Proc. PAM*, 2008.

[42] F. Schneider, A. Feldmann, B. Krishnamurthy, and W. Willinger. Understanding online social network usage from a network perspective. In *Proc. IMC*, 2009.

[43] Z. Li et al. WebProphet: Automating performance prediction for web services. In *Proc. NSDI*, 2010.

[44] Y. Zhang, H. Zhu, and S. Greenwood. Website complexity metrics for measuring navigability. In *International Conference on Quality Software*, 2004.