Empirical Validation of Performance Metrics for Web Page

# Project Guide:

# Mr Manoj Sethi

# Aashna Garg (2K11/SE/001)

# Apoorva Mittal(2K11/SE/010)

## 1. Introduction

**Web metrics** is the [measurement](http://en.wikipedia.org/wiki/Measurement), [collection](http://en.wikipedia.org/wiki/Data_collection), analysis and reporting of [web data](http://en.wikipedia.org/wiki/Data_(computing)) for purposes of understanding and optimizing [web usage](http://en.wikipedia.org/wiki/Web_usage) [28]. Web analytics is not just a tool for measuring [web traffic](http://en.wikipedia.org/wiki/Web_traffic) but can be used as a tool for business and [market research](http://en.wikipedia.org/wiki/Market_research), and to assess and improve the effectiveness of a web site. Web analytics applications can also help companies measure the results of traditional print or broadcast [advertising campaigns](http://en.wikipedia.org/wiki/Advertising_campaign). It helps one to estimate, how traffic on a website changes, after the launch of a new advertising campaign. Web analytics provides information about the number of visitors to a website and the number of page views. It helps gauge traffic and popularity trends which is useful for market research.

Web page metrics is one of the key elements in measuring various attributes of web site. Metrics gives the concrete values to the attributes of web sites which may be used to compare different web pages .The web pages can be compared based on the page size, information quality ,screen coverage, content coverage etc. Internet and website are emerging media and service avenue requiring improvements in their quality for better customer services for wider user base and for the betterment of human kind. E-business is emerging and websites are not just medium for communication, but they are also the products for providing services.

### 1.1 Quality Metrics

Quality of web page can be determined using quality metrics. These metrics can be classified into three categories:

1. Page Composition Metrics: Examples of this metrics are number of words, body text words, words in page title, total number of links etc [28].
2. Page Formatting Metrics: These comprise of font size, font style, screen coverage etc.
3. Overall Page Quality or Assessment Metrics: Examples are information quality, image quality, link quality etc.

Various metrics which help us determine the quality of the web page are :

* *Number of words*

Total number of words on a page is taken. This attribute is calculated by counting total number of words on the page. Special characters such as & / are also considered as words.

* *Body Text Words*

This metrics counts the number of words in the body versus display text that is headers. In this, the words that are part of body are calculated separately from the words that are part of display text.

* *Number of Links:*

These are total number of links on a web page and can be calculated by counting the number of links present on the web page.

* *Page Size:*

It refers to the total size of the web page.

* *Number of Graphics:*

These refer to the total number of images on a page. And can be calculated by counting the total number of images present on the page.

### 1.2 Performance Metrics

Under the average website load scenario, numerous users use their browsers to surf certain websites. A single web page presented in a browser can generate tens, sometimes hundreds of unique HTTP requests. The browser, after receiving all of the responses renders the page for the user to view. Every browser (IE, Firefox, Chrome, etc.) has its own way of generating the HTTP request according to the viewed web page. In a load scenario, the overall number of HTTP requests is directly related to the number of users that are surfing concurrently. All requests hit the server during the test generating a response for each request.

The goal of performance testing is to evaluate the application’s performance with respect to real world scenarios. The issues which must be addressed are that the performance of the system during peak hours i.e, response time, reliability, availability etc., points at which the system performance degrades or system fails as well as the impact of the degraded performance on the customer loyalty, sales and profits.

Two major components of Performance Testing are Load Testing and Stress Testing.

### 1.2.1 Load Metrics

**Load testing is simulating a high volume of concurrent users against a software application. The primary objective is to** measure how well the application performs during heavy traffic. The four major components of load testing are understanding the environment of the application being tested, a test plan user behavior, tools to generate load and capture metrics, and analysis of performance indicators.

### 1.2.2 Stress Metrics

Stress testing involves use f web site with more than maximum and varying loads for a long period. Unlike performance and load testing, stress testing evaluates the response of the system when the system is given a load beyond its specified limits. It is also used to monitor and check the reliability of the site when available resources are on beyond maximum usage. The behavior of the system is monitored to determine when the system under stress test fails and how does it recover from the failure. Stress tests may test the web sites and applications for CPU and memory usage, response time, backend database, difficult type of users, concurrent users etc.

## 2 Literature Review

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 [3, 1] rather than the structure and content of individual websites. Related efforts have studied how the content of individual web pages evolves over time. Recent efforts have also tried to “map” the hosting sites from which content is served [4]. Performance and optimization: As the usage scenarios for the Web have changed, researchers have analyzed inefficiencies in web protocols and suggested improvements [6]. In parallel, there are efforts toward developing better browsers [20], tools to optimize web pages, benchmarking tools [10, 8], services for customizing web pages for different platforms [25], and techniques to diagnose performance bottlenecks in backend infrastructures [26] and to debug client performance in the wild [2].

Web traffic measurement: This includes work on measuring CDNs [17], understanding emerging Web 2.0 and AJAX-based applications [20, 26], measuring the network impact of social network applications [21, 25], and characterizing end-user behavior within enterprises [13], and longitudinal studies [24] 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 [3, 9]. There are also commercial services that measure page load times in the wild. 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. [16] 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 [8], network management [5], and systems more broadly [8]. In a Web context, Zhang et al. [27] present metrics to capture ease of web page navigation and Levering et al. [19] 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 web pages: The closest related work appears in recent industry efforts: HTTP Archive and at Google. 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. Description of Metrics selected for the study

Following metrics parameters are used for the study:

1. **Average Response Time:** When you measure every request and every response to those requests, you will have data for the round trip of what is sent from a browser and how long it takes the target web application to deliver what was needed.

For example, one request will be a web page…let’s say the home page of the web site. The load testing system will simulate the user’s browser in sending a request for the “home.html” resource. On the target’s side, the request is received by the web server, it makes further requests of the application to dynamically build the page, and when the full HTML document is compiled, the web server returns that document along with a response header.

The Average Response Time takes into consideration every round trip request/response cycle up until that point in time of the load test and calculates the mathematical mean of all response times.

The resulting metric is a reflection of the speed of the web application being tested – the BEST indicator of how the target site is performing from the users’ perspective. The Average Response Time includes the delivery of HTML, images, CSS, XML, Javascript files, and any other resource being used. Thus, the average will be significantly affected by any slow components.

**2) Peak Response Time:** Similar to the previous metric, Peak Response Time is measuring the round trip of a request/response cycle. However the peak will tell us what is the LONGEST cycle at this point in the test.

For example, if we are looking at a graph that is showing 5 minutes into the load test that the Peak Response Time is 12 seconds, then we now know one of our requests took that long. The average may still be sub-second because our other resources had speedy response.

The Peak Response Time shows us that at least one of our resources is potentially problematic. It can reflect an anomaly in the application where a specific request was mishandled by the target system. Usually though, there will be an “expensive” database query involved in fulfilling a certain request such as a page that makes it take much longer, and this metric is great to expose those issues.

Typically images and stylesheets are not the slowest (although they can be when a mistake is made like using a BMP file). In a web application, the process of dynamically building the HTML document from application logic and database queries is usually the most time intensive part of the system. It is less common, yet occurs more often with open source apps, to have very slow Javascript files because of their enormous size. Large files can produce slow responses that will show up in Peak Response Time, so be careful when using big images or calling big JS libraries. Many times, you really only need less than 20% of the Javascript inside those libraries. Lazy coders won’t take the trouble to clean out the other 80%, and that will hurt their system performance.

1. **Latency Time:** It measures the time between which the request was sent till the first byte is received.

The above parameters are measured in terms of seconds.

1. **Throughput:** Throughput is the measurement of bandwidth consumed during the test. It shows how much data is flowing back and forth from your servers. It is measured in terms of bytes and can be determined by the total size of the page in bytes.
2. **Hits:** This measure represents the number of hits per minute at a certain minute of the test. This tells the successful access to sites in a given minute.

There are metrics which help to determine the failures for example error rate which represents the parentage of problem requests to all requests. The percentage reflects how many responses are HTTP status codes indicating an error on the server, as well as any request that never gets a response.

No one can define the tolerance for Error Rate in your web application. Some testers consider less than 1% Error Rate successful if the test is delivering greater than 95% of the maximum expected traffic. However, other testers consider any errors to be a big problem and work to eliminate them. It is not uncommon to have a few errors in web applications – especially when you are dealing with thousands of concurrent users.

## 3. Experiment

The project focuses on performance testing and load testing where a number of metrics are chosen to compare different websites. A tool was created using python which extracts all the required information and parameters. For the purpose, we have used Multi Mechanize, which is an open source framework used for the same. It runs concurrent Python scripts to generate load which are synthetic transactions against a remote site or service.

Any number of “user groups” can be specified, each representing one sequence of actions on your web site/service (e.g. loading the home page, or executing a search). Each user group is associated with a python script defining the actions. Each user group has a number of threads, i.e. virtual users attempting the sequence of operations.

The platform is simulated for fifty users concurrently accessing the web site under test.

The data for which the analysis and performance testing is carried out constitutes of 40 educational websites with varied amount of data in form of text, images, videos etc.

### 3.1 Software Specification:

Python IDE, version 2.7 is used with the support of modules like mechanize, urllib, urllib2, httplib etc. which are run on Windows OS, intel core i5. This tool can be made portable to other platforms like Mac iOS or linux.

### 3.2 Hardware Specification

Other than a laptop, no specific hardware is required.

Given below is the table which records the values of the parameters for each website run under identical environment and load.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Web Site** | **Response Time(s)** | **Peak Response Time(s)** | **Latency Time** | **Throughput(bytes)** | **Number of Hits** |
| http://academicearth.org/online-college-courses/engineering/ | 4.778 | 7.02 | 0.812 | 114273 | 74 |
| http://bigthink.com/ | 7.951 | 20.81 | 0.874` | 67261 | 69 |
| http://www.brightstorm.com/ | 4.457 | 7.84 | 0.79 | 31319 | 76 |
| http://www.cosmolearning.com/ | 6.69 | 10.23 | 1.64 | 92081 | 37 |
| https://www.coursera.org/ | 3.176 | 12.55 | 2.5 | 6946 | 24 |
| https://www.edx.org/ | 5.38 | 13.01 | 2.18 | 54702 | 28 |
| http://thefutureschannel.com/ | 3.41 | 8.11 | 1.367 | 33093 | 44 |
| http://www.howcast.com/ | 1.45 | 1.6 | 0.73 | 33425 | 82 |
| <https://archive.org/details/computersandtechvideos> | 6.48 | 10.39 | 3.07 | 85353 | 20 |
| http://www.apple.com/apps/itunes-u/ | 0.54 | 1.13 | 0.33 | 13557 | 182 |
| https://www.khanacademy.org/ | 4.98 | 6.69 | 2.25 | 109835 | 27 |
| <http://learner.org/resources/browse.html> | 3.55 | 4.72 | 1.05 | 102615 | 57 |
| http://www.mathtv.com/ | 3.25 | 3.88 | 1.68 | 62459 | 36 |
| <http://ocw.mit.edu/index.htm> | 2.24 | 11.59 | 0.52 | 48763 | 115 |
| http://video.mit.edu/ | 3.34 | 4.12 | 1.08 | 73790 | 56 |
| http://www.neok12.com/ | 1.42 | 2.53 | 0.76 | 16246 | 79 |
| https://www.youtube.com/user/ResearchChannel/ | 8.69 | 15.09 | 1.26 | 165283 | 48 |
| http://video.pbs.org/ | 5.91 | 13.53 | 1.84 | 43735 | 33 |
| http://www.schooltube.com/ | 2.81 | 3.75 | 0.86 | 63124 | 70 |
| http://www.schoolsworld.tv/ | 3.67 | 4.27 | 2.164 | 64255 | 28 |
| http://www.teachertube.com/ | 4.82 | 5.73 | 3.56 | 99744 | 17 |
| http://www.ted.com/talks/browse | 3.20 | 11.51 | 1.27 | 71673 | 47 |
| http://www.videojug.com/ | 2.189 | 2.46 | 0.897 | 56841 | 67 |
| http://www.watchknowlearn.org/ | 5.44 | 7.8 | 3.354 | 140346 | 28 |
| http://www.wonderhowto.com/ | 4.41 | 6.85 | 1.034 | 285085 | 58.03 |
| http://oyc.yale.edu/ | 1.29 | 1.40 | 0.90 | 27008 | 67 |
| https://www.youtube.com/channel/UC3yA8nDwraeOfnYfBWun83g | 5.70 | 8.51 | 1.26 | 570584 | 48 |
| http://www.brainpop.com/ | 0.59 | 085 | 0.57 | 19014 | 105 |
| <http://www.cosmeo.com/welcome/index.html?CFID=25759495&CFTOKEN=34090996> | 0.92 | 2.15 | 0.83 | 6511 | 72 |
| https://vimeo.com/ | - | - | - | 30673 | - |
| http://nptel.ac.in/ | 2.36 | 4.71 | 0.70 | 56812 | 86 |
| https://www.udacity.com/ | 2.5 | 2.81 | 2.06 | 27358 | 29 |
| https://learni.st/ | 1.40 | 2.27 | 1.44 | 2196 | 42 |
| http://www.memrise.com/ | 1.13 | 1.31 | 0.73 | 28516 | 82 |
| www.mentormob.com/beta/splash | 2.88 | 5.26 | 1.28 | 10684 | 47 |
| http://ed.ted.com/ | 1.95 | 2.33 | 1.59 | 40838 | 38 |
| http://www.edutopia.org/ | 3.53 | 4.19 | 1.59 | 60527 | 38 |
| http://www.thinkfinity.org/welcome | 10.62 | 24.14 | 2.27 | 198372 | 26 |
| http://www.goorulearning.org/#discover | 2.53 | 5.93 | 0.77 | 5606 | 78 |
| http://www.starfall.com/ | 1.34 | 1.41 | 0.78 | 15519 | 77 |

## 4. Result and Analysis

The regular pattern above is that as the size of the page increases in terms of bytes that is the throughput then the latency time increases and number of hits per minute decrease. The peak response time can be very high in some cases even though the average response time is normal hinting at an element which took large time to load. For example in the above table, for http://www.ted.com/talks/browse, the average response time is 3.2 s but the peak response time is 11.5 s.

Also for some sites, even though the throughput is less, the latency time and response time, both are high, which hints at the idea, that there are some other performance metrics which should be considered and analyzed. Also to increase the number of hits, it is important to reduce the latency time as latency is inversely proportional to hit rate.

Also a peculiar example is seen in terms of http://vimeo.com as it could not load and always displayed HTTP error 500 which is an Internal Server Error. Other than that the data behaved in the way it was supposed to.

## 5. Conclusion and Future Work

There is a growing interest in software tools that can analyze code as it is being written. In this project, a static analysis tool based on software metrics, has been presented. The most important feature of the tool is its extendibility: it is straightforward to add new metrics simply by writing a script using Python language. In this way, a metrics researcher needs to deal with a high level, metrics-oriented, intermediate representation when adding or modifying metrics to calculate. This spares the researcher from having to know intimidating language parsing concepts such as declaration, class specifiers, and base clauses. Future enhancements of the system will include rule based analysis in order to check the code for adherence to best programming practices. The integration with most widely used IDEs such as Eclipse, JBuilder, Netbeans, Microsoft Visual Studio, will provide another relevant enhancement. A parser for the C# language is under development and it will be integrated into a future version of the tool.

## 6. References

[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.

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

[2] 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.

[3] 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.

[4] 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.

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

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

[7] 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.

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

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

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

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

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

[13] 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.

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

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

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

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

[18] 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.

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

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

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

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

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

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

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

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

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

[28] Yogesh Singh, Ruchika Malhotra. Empirical Validation of Web Metrics for Improving the Quality of Web Page. International Journal of Advanced Computer Science and Applications.