Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 2, February 2014, pg.834 – 855

RESEARCH ARTICLE

A TECHNIQUE FOR IMPROVING CAKEPHP WEB PAGE RESPONSE TIME

Pharaoh Chaka[1], Hilton Chikwiriro[2]

[1]Computer Science Department, Bindura University of Science Education, Zimbabwe

[2]Computer Science Department, Bindura University of Science Education, Zimbabwe

[1] pchaka01@gmail.com; [2] hchikwiriro@gmail.com

Abstract— Research on web performance has mainly concentrated on the network medium over which pages are transmitted. While it is quite clear that it is the speed of transmission that has the greatest effect on the perceived page load time on the client, this research seeks to evaluate the effect of web development practices particularly the use of PHP frameworks on the response time of web pages. The goal is to find a way that can significantly reduce the response time of framework-based web pages. The technique to be considered by this research is the use of AJAX technology to load dynamic content. In this proposed technique, when a user accesses a website for the first time in a session, a general template (consisting of HTML and CSS files) of the page is sent to the browser and displayed before the dynamic content starts loading. This technique uses the general idea that once a user sees something appearing on the screen they become patient and wait for the rest of the page to load.

Keywords— web performance; web development; PHP frameworks; response time; AJAX

I. INTRODUCTION

In a rapid changing environment, software applications must be delivered on time before business delivery deadlines. Simple value-for-money systems that work are better than expensive and complex ones delivered late, over-budget and difficult to maintain (Choo & Lee, 2008).Rapid Application Development (RAD) promises software development teams a very short development lifecycle. The idea is to develop a method of designing software so that the whole process is quick, painless and nearly effortless. The tools used should be easy to learn, powerful and allow the designers to interface their freshly minted application with other applications, databases and file types. Ideally, the use of these tools must also not compromise the performance of the end product. Web development is one area of software development where RAD is the model of choice. The use of frameworks in developing websites is a very common phenomenon among web developers. However, though frameworks drastically reduce the development lifecycle time, they produce websites that perform badly compared to hand coded sites in most cases. There are many metrics for measuring performance of web pages, and user satisfaction (usability of the page). One important metric is the response time of the page i.e., the time it takes from the moment the user requests the page until that page is loaded in the client browser. This research thus focuses specifically on a PHP RAD tool (or framework) called CakePHP and its effect on the response time of web pages.

II. WEB APPLICATION DEVELOPMENT

Web application development is the design, coding, and maintenance of software systems meant to be accessed over the World Wide Web. These systems are quite different from the traditional desktop applications which are installed on each individual computer.

A. Characteristics

One major characteristic of web application development is the fact that the development teams tend to be rather small in number and made up mostly of young multidisciplinary people who lack a certain amount of experience (McDonald and Welland 2001a). On the other hand they are more open to new technologies and are eager to explore new ways of doing things. Web applications also tend to suffer from immaturity as the time-to-market is so brief (Ziemer 2004), resulting in incremental releases in order to reach the desired functionality.

Another interesting phenomenon of web development is the abundance of development communities who provide help and also develop components and fully-fledged systems that can be reused free of charge (Haddad 2007). Thus many programmers include these components during development as a way of speeding up the development process. However these components may contain some flaws which are overlooked and prove to be problematic later. Flexibility in web development is of outmost importance as a rigid project plan is usually impossible to follow.

A major defining characteristic for web applications that distinguishes them from traditional desktop applications is the fact that web applications are hosted on a remote server, from where end users use client machines to access the content on the server. Thus updating web-based applications is much easier than updating copies of desktop applications. Clients always have access to the latest version of the web application as on the server. However, on the down side, access to these web-based applications is usually very slow and in most cases negatively impacts the user experience. In recent years, a new approach has been applied to web development that increases the responsiveness of web applications. This technology is known as Asynchronous JavaScript and XML (AJAX).

B. AJAX

The name is an acronym for Asynchronous JavaScript and XML and was first coined by Jesse James Garrett. Ajax is not a technology. It is really several technologies coming together in powerful new ways. Ajax incorporates:

- Standards-based presentation using XHTML and CSS;
- Dynamic display and interaction using the Document Object Model;
- Data interchange and manipulation using XML and XSLT;
- Asynchronous data retrieval using XMLHttpRequest;
- JavaScript binding everything together.

The classic web application model works like this: Most user actions in the interface trigger an HTTP request back to a web server. The server does some processing — such as retrieving data, or computing some calculations — and then returns an HTML page to the client. It is a model adapted from the Web's original use as a hypertext medium, but as the Web is being transformed from just a hypertext medium to a software application medium, the model has lost touch. The model does not take into consideration the user experience because while the server is building up the response, the user is just staring at a blank screen, waiting for the page to load in the browser.

When an Ajax request is made to a server, the currently displayed page is not unloaded from the web browser but rather only the affected CSS div element is updated. This element is usually just a small section of the page and it is updated asynchronously without refreshing the whole page. In addition, by using the XMLHttpRequest object's onreadystatechange attribute it is possible to inform the user about the status of their request. This greatly improves the user experience as users do not like being kept waiting without knowing what is going on behind the scenes.

C. Web Development Methodologies

Due to the differences between desktop software projects and web projects, Kappel (2006) argues that many concepts, techniques, and tools of traditional software engineering have to be adapted to the needs of web engineering and might also prove to be totally inadequate.

Currently there are two main types of methodologies used. The first is the 'older heavyweight methodology' which is characterised by comprehensive planning, thorough documentation and expansive design (Khan, 2004). The second type of methodology is called a 'lightweight (or agile) methodology' and attempts to minimize risk and maximize productivity by developing software in short iterations and de-emphasizing work on secondary or interim work artefacts.

(McDonald & Welland, 2001b) put the average web application development lifecycle at typically 3 months or less. The agile methodology is therefore more suited to web application development because the requirements for web based applications are ever changing. The rapidly evolving web technologies also necessitate short development lifecycles for web applications. It would not be desirable to release an application which relies on a deprecated version of a particular technology. For example, the PHP scripting language is always under development with new versions being released regularly. It is thus favourable for any new web applications to always use functions supported by the latest release of PHP.

Rapid application development is an agile methodology that was a response to the non-agile processes.

D. CakePHP

CakePHP is a free open-source rapid development framework for PHP. It is a structure of libraries, classes and run-time infrastructure for programmers creating web applications originally inspired by the Ruby on Rails framework. Its goal is to enable the developer to work in a structured and rapid manner - without loss of flexibility. (CakePHP Manual, 2010)

CakePHP follows the Model-View-Controller (MVC) architectural pattern to separate the data model with business rules (controller) from user interface (view). This is generally considered a good practice as it modularizes code, promotes code reuse, and allows multiple interfaces to be applied. It also permits independent development, testing and maintenance of each.

III. Response Time

There are many metrics to be used in the analysis of a system's performance. Probably the most often used metrics for a web-based system's performance analysis are the response time and throughput. Since the success of web-based systems depends mostly upon user satisfaction, the best metric to study the performance of a web-based system should be one that describes system performance from the users' perspective. (Broadwell, 2004) says that one metric that impacts all users is response time, while Neilsen (2000) is quoted: "...fast response times are the most important design criterion for web pages."

Response time, or latency, is the time elapsed between two related events-- a start and a stop event. In respect to web applications, there are two definitions of response time from the user's perspective, differentiated by the descriptions of the stop event. The definitions are:

a. The time elapsed from the moment the user requests a Web page until the requested page is displayed in its entirety on the user's machine.

b. The time elapsed between the start of the request and beginning of the response, i.e. the page, starts displaying on the user's machine. This is the definition adopted by this research.

Regardless of which definition is adopted, times for DNS lookup, TCP connection set up, requesting and retrieving the Web page and its embedded objects will constitute user-perceived response time. This is a view of response time components from the chronological order of the activities that make up a transaction. However, with the new proposed technique, the time for requesting and retrieving embedded objects does not constitute the response time as defined in b) above. Spatial-wise, response time can be decomposed into three constituents, namely:

- Network transmission,
- Server-side request processing and response generation, and
- Client-side request sending and response displaying.

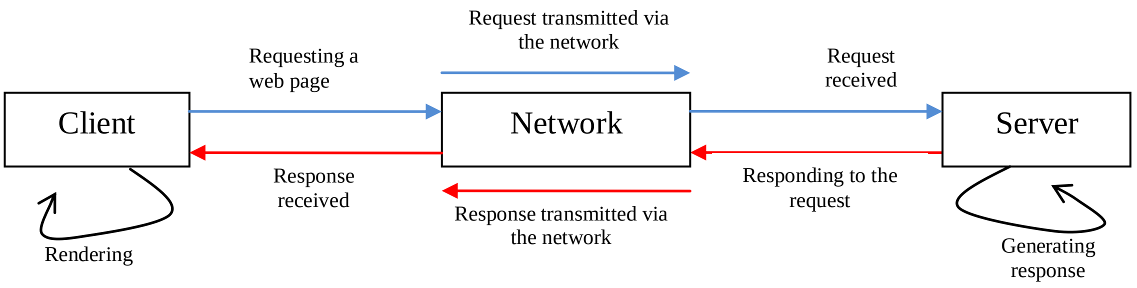


Figure 3.1: Three Parties Involved in Retrieving a Web Page

Decomposing response time into its constituents helps us to identify and isolate possible bottlenecks of Web performance.

One of the issues about response time that draws the attention of Web suppliers, especially e-Commerce practitioners, is how long the users are willing to wait for a Web page to be downloaded before giving up.

Nielsen (1999) put a very high regard on the importance of response time or download speeds as “the single-most important design criterion on the Web”. Rose, et al., (1999) ranked download time as the second most important Technological Impediment to Electronic Commerce (TIEC). Similarly, a panel of experts participating in a Delphi study ranked download delay as the single most worrisome issue “affecting the overall utilization and management of Web-enabled technologies” (Khosrowpour and Herman, 2000, p. 1). In a test of 63 respondents, Simon Galbraith and Neil Davidson found page load times between 6 and 15 seconds had high bailout rates and caused significant frustration among users. "Any wait beyond that is highly likely to cause the user to abandon the site." (Davidson & Galbraith, no date)

In addition Hoxmeier and DiCesare (2000) found that user satisfaction is inversely related to response time. They said that response time "could be the single most important variable when it comes to user satisfaction."

There seems thus to be consensus on the fact that response time is the most important metric for measuring the performance of web applications, however the only source of contention is how long the maximum acceptable response time should be. In this regard, there are many differing views on the specific value. For instance, over a decade ago, Nielsen (1997) had proposed ten seconds as the maximum response time for downloading a Web page. Zona (1999) proposed a lower threshold of eight seconds. Wikipedia (2010d) offers the following guidelines for ideal Web response times:

- 0.1 second (one tenth of a second). Ideal response time. The user doesn't sense any interruption.
- 1 second. Highest acceptable response time. Download times above 1 second interrupt the user experience.
- 10 seconds. Unacceptable response time. The user experience is interrupted and the user is likely to leave the site or system.

These figures are however just rough guidelines of the average tolerable waiting time and cannot be used as benchmarks in all situations. This is because it has been noted that user perceived web latency is not only measured by the actual response time but also affected by psychological factors of the user, such as tiredness. The Web-surfing context such as the type of Web-based system or application, the user’s Internet connection speed, and types of tasks carried out also affects the user’s tolerable waiting time (Nah, 2003). For example, comparing simple information retrieval tasks and online purchasing, it has been found that users are more willing to wait longer for the latter as they have a more vested interest in it (Kotsis, 2006). It is also commonly believed that giving hints or information about waiting time in advance to the users, or retrieval information and status given while the users are waiting for Web pages to download, will increase the users’ willingness to wait (Hitz et al., 2006).

A. Measuring Response Time

Response time measurement methods for a web site can be grouped into three categories:

- Active probing,
- Server-side measurement, and
- Client-side measurement

Active Probing

With active probing, a few geographically distributed synthetic clients, called the agents, are used to periodically probe the server by requesting a set of Web pages or operations. The agents mimic users from different locations over the world. The measurements obtained are representations of latencies that may be experienced by the end users. Active probing therefore produces real world results from faked end users. In order to produce results representative enough of the massive Internet traffic from largely diversified users across the world with varied connection speeds, active probing requires machines with different capabilities to be setup in many different locations and large amounts of measurement to be taken daily. One of the examples of active probing is Keynote (http://www.keynote.com), a commercial provider of test and measurement products for Internet performance. More than 100, 000, 000 Internet performance measurements are taken daily.

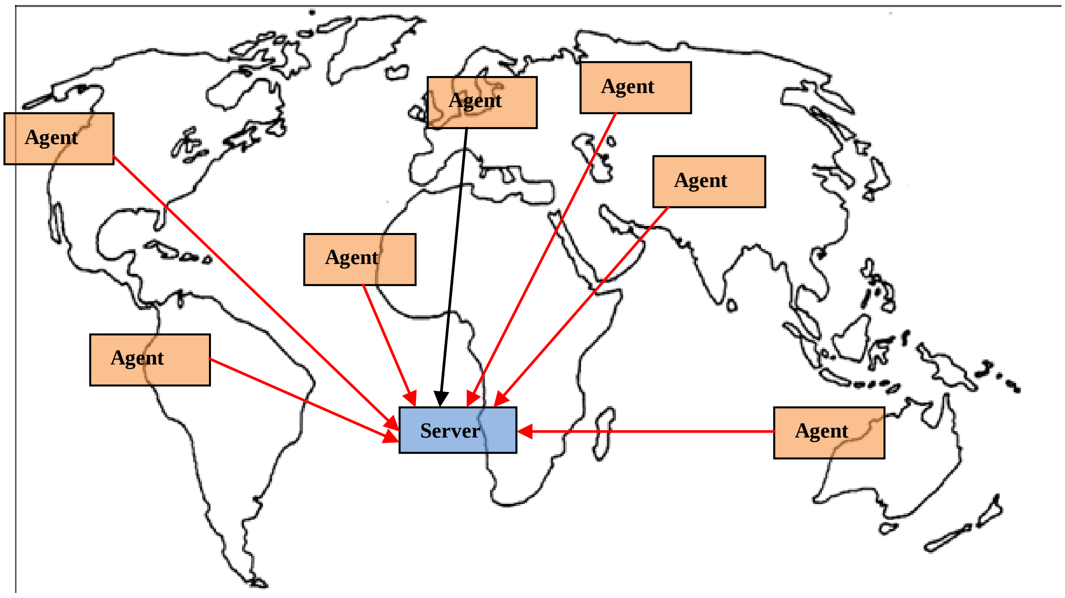


Figure 3.2: Active Probing: Agents Distributed across the World Actively Probing the Server to Perform Measurements

Server-side Measurements

As the name implies, server-side measurement performs measurements of various performance metrics at the Web server. Server log analysis is the most commonly used method. A Web server log contains fields that describe each request a browser makes from the server (Rosenstein, 2000). The fields include:

- The IP address or the host name from which the request originated.
- The date and time of when the request was made.
- The request line that shows the method by which the resource was requested (such as GET or POST), the resource requested together with the protocol used by the client (such as HTTP/1.0 or HTTP/1.1).
- The status code that indicates if the request resulted in a successful response (codes beginning in 2), a redirection to another resource (codes beginning in 3), an error caused by the client (codes beginning in 4), or an error in the server (codes beginning in 5).
- The size of the object returned to the client excluding the response headers.
- The browser or user agent field that indicates the browser used by the user to make the request.
- The referrer field that indicates the page from which the user made the request.
- Different information can be extracted from these fields to fulfil the interests of different parties. For example, the “IP address” field can be used to show the geographic distribution of visitors to the web site while the “user agent” field tells which browser was used by most of the visitors. Even though the server log could provide a large amount of data with rich attributes that can be transformed into appropriate and easily assessable actions (Kohavi, 2001), it nevertheless has difficulty to produce desirable information from log files. Since the server log only records individual requests and downloading a web page may yield a few non-consecutive entries in the log, it does not provide direct information about the time taken to download a full page, unless heuristics are used to identify sessions from the log entries. Furthermore, if a number of clients are situated behind the same proxy server, they will appear to have the same IP address (of the proxy server) in the server log. Apart from that, if the client requests and retrieves Web content that is cached, the requests won’t appear in the server log.

Despite the above-mentioned problems, server log analysis is still a favourite method for performing measurements, probably because of its availability. One of the reasons is that it does not introduce traffic into the network. Other advantages include the fact that it can evaluate each client’s experience without the need for instrumentation at every client or additional agents placed on the Internet, and implementation at the HTTP level rather than lower protocol levels (Marshak & Levy, 2003). Server side measurements also reduce concerns about intruding on users’ privacy, which is a problem with instrumented page. To overcome the problem related to the absence of session information in the server log, Krishnamurthy and Wills (2002) proposed the following heuristics to extract information from the server log and induce a sequence of requests that make up the download of a web page:

- The first request by a client, indicated by the client IP address or host name, returns an HTML object or the output of a server side script, called the base object.
- Each subsequent request from the client is for an image, style sheet, or JavaScript object.

- If the referrer field is set, then the referrer field for the embedded objects must also match the URI of the base object.
- An arbitrary maximum threshold of 60 seconds is defined between the time the base object was downloaded and any subsequent requests. Any subsequent requests made beyond this threshold are not classified as making up the original web page. The value of this threshold is a significant factor that determines the accuracy of the approximation of response time.

Another approach used by Marshak and Levy (2003) was to add a tiny inline HTTP object called the sentry (better known as a Web bug) at the end of the HTML document. They assumed that the request for the sentry will be the last corresponding entry recorded in the server log for a Web page downloaded. However, this assumption is flawed because HTTP does not require embedded objects to be requested in the sequence as they appear in the HTML document. Thus, the sentry is not necessarily the last object to be requested even though it is placed at the end of the document.

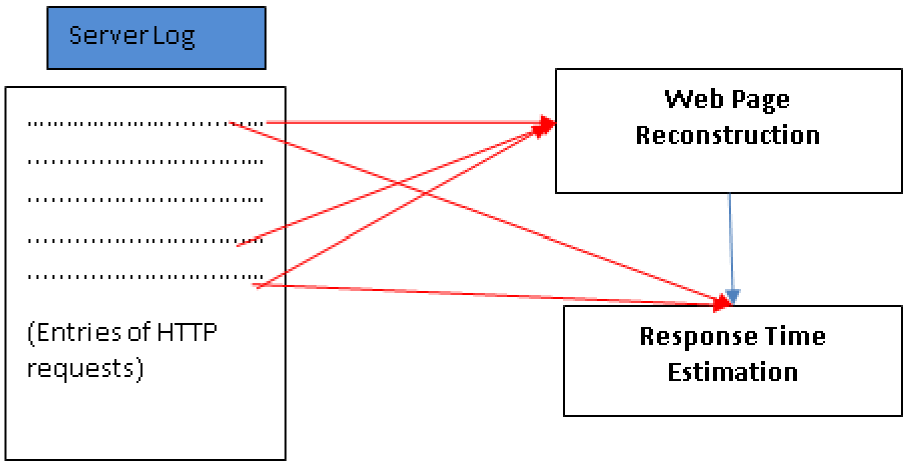


Figure 3.3: Using Server Log Analysis to Estimate Response Time

Besides server log analysis, server-side measurement methods also include measurements made at lower protocol levels. For example, Cherkasova et al. (2003) proposed a tool called EtE monitor to measure response time by passively collecting packet traces from a server site. The EtE monitor architecture is shown in Figure 3.4.

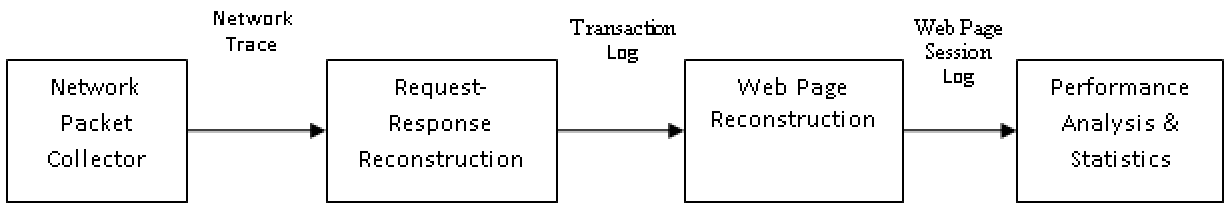


Figure 3.4: EtE Monitor Architecture (Cherkasova et al., 2003)

The network packet collector module collects network packets using tcp-dump5 and records them to a network trace for offline analysis. The request-response reconstruction module uses the network trace to reconstruct all TCP connections using client IP addresses, client port numbers, and request-response TCP sequence numbers. HTTP transactions are then extracted and the HTTP header lines of each request recorded in the transaction log for future processing. The web page reconstruction module groups underlying physical object retrievals together into logical web pages and stores them in the web page session log. The Web page reconstruction process is similar to that achieved by extracting information from the server log to identify requests that make up a web page. Analysis thereafter could be done based on the reconstructed web page. Again, the accuracy of the reconstructed Web page relies on the heuristics used and the amount of data available for inferring web pages from raw data.

Client-side Measurements

Client-side measurements make use of instrumentations such as scripting languages or specialised software at the client side to acquire the desired information. Cookies might be used to record information at the client side. A cookie is a small parcel of information issued by the web server to a web browser to uniquely and anonymously identify the user using that particular browser.
Rajamony and Elnozahy (2001) used JavaScript to instrument hyperlinks in a set of web pages for measuring response time. When an instrumented link is activated, the current time is determined and remembered, and then the request is sent to the web server. After the requested page and its embedded elements have been fully loaded to the browser, the client browser computes the response time as the difference between the current time and the previously stored time. The response time can be transmitted to a record-keeping web site on a different server from the originally responding Web server for further analysis. Note has to be taken that the time samples taken need to be stored in cookies, a dedicated window or a frame within a browser window as the browser cannot maintain the values across page loads any other way. Figure 3.5 depicts this approach, which allows accurate measurement of response times as experienced by the end user.

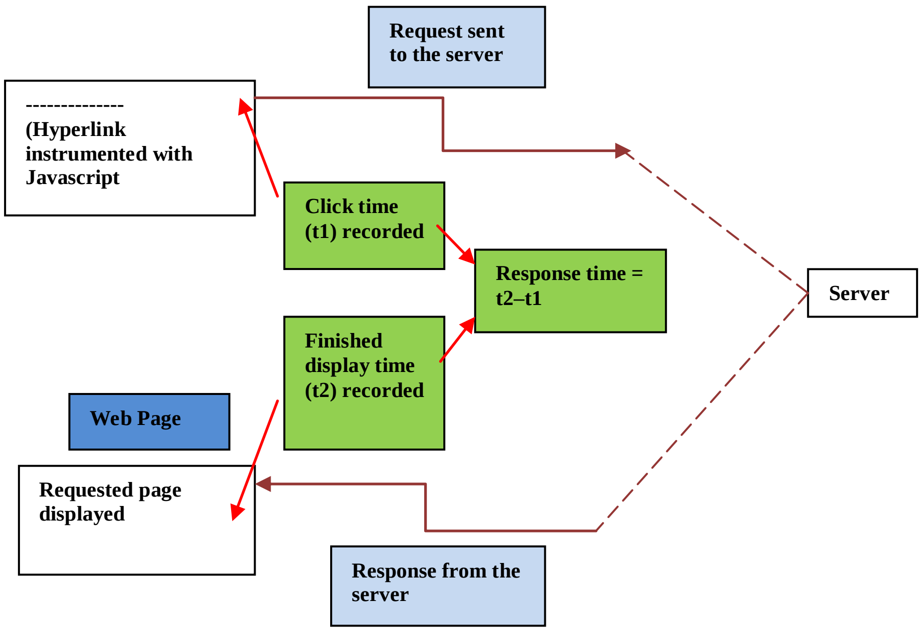


Figure 3.5: Client Side Measurement of Response Time using JavaScript Instrumented Links

There are however, four problems with this approach. Firstly, the approach does not compute response time for a request which is not made through an instrumented hyperlink. Secondly, pages containing images or PDF files cannot be instrumented. Thirdly, the approach does not work with browsers that do not support the scripting language used for instrumentation, or when the execution of the scripting language is disabled by the user. Fourthly, the cookie itself impacts response time.
Clickstream, a company that collects, transforms and transmits information on its customer and commercial activities on Web sites or any other HTTP environment (http://www.clickstream.com), uses third party cookies for measuring response time as well as page view time and tracking user activities when surfing the monitored pages. Cookies are used to deal with the problem of HTTP being a stateless protocol. Cookies are able to keep track of user sessions. Thus, communication between the client and the server, as well as the user activities on the browser, can be detected and recorded by the cookies. Clickstream installs a software module on the master content server. The software module embeds tracking code into the header of each Web page dispatched by the server. The module also manages and processes tracking data that is returned to the server as a tracking cookie from the Web browser. Even though Clickstream claims to be a non-intrusive and privacy friendly system, as long as cookies are used, the concern about the line between acceptable monitoring and intrusion of user privacy will always exist. Moreover, this method only works when cookies are enabled at the user's browser.

B. Improving Response Time
In spite of many factors that affect user perceived web latency, response time has the closest and the least arguable relationship with users' perception of the delay. Improving web response time is the most

straightforward way of improving users' perceptions of a particular web site's latency. In order to improve response time, different approaches can be adopted. A discussion of the most common approaches follows.

Caching

Caching is the most important and most widely used performance improvement technique for web-based systems. The idea of caching is to keep frequently accessed data at locations close to the clients such as the client browsers or web proxy servers. Retrieving data from these caching locations will not only reduce transmission time across the internet, but also reduce workloads imposed on the web server. Thus, caching trades storage space and currency of content for access speed.

The main issue with caching is to maintain consistency of the cache across the web. It is a less serious problem with static web content but for dynamic web content, caching may deliver outdated information. There are four categories of degrees of consistency between the content at the web server and those cached, i.e. strong, delta, weak and mutual consistencies (Iyengar et al., 2002). The four categories are:

a) Strong consistency. The latest content is always returned by the cache. This requires the client to poll the server to validate the currency of cached objects should they be requested, or the server invalidates or updates all the outdated cached objects even though they are not requested. However, exchanging of messages between the client and the server requires time within which the object may have changed again after the server validates its currency. Message delay on the internet is also unbounded, making strong consistency hard, if not impossible, to guarantee. Therefore no cache consistency mechanism can be ideally and truly strong.

b) Delta consistency. Data returned by the cache is never outdated by more than δ time units, where δ is a configurable parameter. The value of δ should be larger than the network delay between the cache and the server. If the value of δ is the same as the network delay or slightly greater than it, delta consistency can therefore be regarded as the practical implementation to achieve strong consistency.

c) Weak consistency. A read at the cache may retrieve some previously correct content.

d) Mutual consistency. A group of objects are mutually consistent with respect to each other.

Web Standards

Web standards are formal technical specifications that define and describe different aspects of the web. Web standards ensure that web-based content can be created and interpreted correctly by different browsers as well as ensuring the content is accessible via different platforms and devices. Some web standards are also relevant to response time. One such standard is HTTP/1.1. Prior to HTTP/1.1, each request made by the client required a TCP connection to be established between the client and the server that served the request. Since a web page will typically consist of an HTML file as the container and a few embedded objects, such as images, downloading a web page would involve the client making more than one request to the server and each request would involve establishing a TCP connection. According to Chandranmenon and Varghese (2001), setting up such a connection may take up to 80ms.

With HTTP/1.1, persistent connection and pipelining are supported. Persistent connection allows a TCP connection to be re-used to retrieve multiple objects from the same IP address. Pipelining, on the other hand, allows the client to make a series of requests over the same connection without waiting for the completion of the previous response. Both persistent connection and pipelining will reduce response time for downloading web pages. Studies from Cherkasova et al. (2003), and Krishnamurthy and Wills (2000) show that HTTP/1.1 does have a positive effect on reducing response time.

Customisation

Web content can be customised to suit the needs and capability of different clients or based on the network traffic condition. Steinberg and Pasquale (2002) used dynamically deployable software modules called Web Stream Customizers (WSCs) located between web clients and servers to adapt web content as necessary to improve performance, including response time, reliability and security of the web service. Their web middleware architecture is especially useful for client machines with limited resources such as wireless palmtops. WSCs provide adaptive system-based and content-based customisation to achieve its goals. The adaptive customisation includes removing data that users are not interested in, filtering images to smaller representations, and displaying Web pages in an easy-to-surf format. Web Content may be compressed or encrypted before it is transmitted onto wireless link which generally has lower bandwidth, and then decompressed or decrypted at the client if the client is powerful enough to deal with this. Compression and encryption can even be an optional step depending on system conditions or user behaviour. For example, compression and encryption may be done only when network throughput is low enough or until the overhead of compression and encryption outweigh delay caused by network congestion.

Such web middleware architecture not only supports content customisation dynamically, but also supports existing web servers, clients, and structures seamlessly, without the need to modify them, in order to achieve better response time. The overhead of the customisation is also low compared to the real web transaction times (Steinberg & Pasquale, 2002).

Server Tuning

There are some configuration parameters related to web servers that can be tuned in order to improve the performance of the server and consequently impact on response time. Apache Web server, for example, has parameters like MaxClients and KeepAlive which may be adjusted to affect server performance metrics like CPU and memory utilisation. Instead of tuning these parameters manually, Diao et al. (2003) proposed a mechanism to automatically tune MaxClients and KeepAlive parameters for the Apache Web server at run-time in order to achieve the desired CPU and memory utilisation. This relieves the system administrator from the tedious, time-consuming, error-prone, and skill-intensive manual adjustment of the parameters. MaxClients defines the maximum number of worker processes that are responsible for communicating with the clients. A higher MaxClients value therefore allows the Apache Web server to process more client requests concurrently and results in higher CPU and memory utilisation. KeepAlive specifies the maximum allowed time for a persistent connection between the client and the server to remain open. Decreasing the value of KeepAlive makes the worker processes more active and thus implies higher CPU and memory utilisation. This may seem to be not directly related to reducing response time, but it does help to keep the Web server at an optimum load level.

Web Technologies (Software)

Many software technologies are involved in different aspects of the Web, including client and server Operating Systems, Web browsers, server software, databases, middleware architectures, and scripting engines and languages. There are some relevant performance tips given by (Killelea, 2002):

- The Web browser seldom becomes the bottleneck that causes lengthy response times but its settings may well be tuned to achieve slightly better performance. For example, not verifying cached pages from the server makes retrieving of the cached pages faster even though it risks the user viewing out of date pages.
- On identical PC hardware, Linux generally gives better performance as a Web client than Windows.
- Unix is more stable and has better performance than other server operating systems because of its longer development history and open nature.
- To improve database performance, actions can be taken including the use of precompiled SQL statements called prepared statements, cache the results of the most frequently used queries, and use a connection pool rather than setting up a connection for each database query.

Since dynamic web content has become an essential component of many web sites nowadays, different aspects related to dynamic web content are studied. One of the aspects is technologies for generating dynamic web content. Cecchet et. al (2003) studied three middleware architectures used for generating dynamic Web content, namely Hypertext Preprocessor (PHP), Java servlets, and Enterprise Java Beans (EJB). The architectures use different mechanisms for generating dynamic web content.

- PHP scripts are tied to the Web server and require the writing of explicit database queries.
- Java servlets execute in a different process from the Web server, i.e. the Java Virtual Machine (JVM). This allows Java servlets to be located on a different machine from the Web server for better load balancing. Database queries are written explicitly, but the Java synchronisation primitives can aid in the communications with the database.
- EJB is a server side component that is used to abstract application business logic from the underlying middleware. An EJB server manages several EJB containers that in turn manage enterprise beans contained within them. The enterprise beans provide different yet common services such as database access, transaction management, messaging, and naming services.

The three architectures were tested on two common application benchmarks: an online bookstore and an auction site. Even though measurements are made in terms of server throughput in number of interactions per minute, and the percentage of server CPU utilisation, the measurements do give some indications of how well these different middleware architectures are at handling client requests and how response time might be affected. As the number of client requests increases, server load will increase too, and so do throughput and response time. Overall, PHP scripts are the most efficient of the three architectures. However, PHP scripts provide limited yet insecure functionality and runtime support. Java servlets can be offloaded to another machine for better performance if the Web server is the bottleneck. Servlets also help to resolve the database lock contention problem if they are the only application that accesses the database. EJB offers the most flexible architecture and supports good software engineering qualities such as modularity, portability, and maintainability. Unfortunately the performance of EJB is the worst among the three architectures.

Load Balancing

Web servers need to handle many requests concurrently and therefore need to perform multithreading or multitasking in order to achieve parallelism. Additional parallelism can be achieved by using multiple servers in conjunction with a load balancer. The function of a load balancer is to distribute requests among the servers. One method of load balancing requests to servers is via DNS servers. DNS servers will translate, or resolve, the clients' requests into one of the associated IP addresses. A particular IP address will be chosen based on policies such as simple round robin, or server load information such as the number of requests received per unit time, as well as network geographic information.

One of the problems with load balancing using the DNS server is that name-to-IP mappings resulting from a DNS lookup may well be cached. The client requests could therefore bypass the DNS server logic and go directly to a server based on the cached information. This will cause load imbalance among the servers. To overcome this problem, the DNS server could assign a time-to-live (TTL) value to each name-to-IP mapping to indicate the duration after which the mapping becomes invalid. Using small TTL values can limit server load imbalance but increase DNS server load and response time, and vice-versa. Adaptive TTL algorithms, on the other hand, assign different TTL values for different clients. Smaller TTL values are assigned to frequently requesting clients compared to clients with low request rates.

IV. RESEARCH METHODOLOGY

A. *Research Instruments*

Software

The following software tools will be used to measure the response time of each web page:

- Mozilla Firefox 3.6 – an Open Source standards-compliant web browser was used for viewing the web pages. Firefox was chosen because it has an add-on component developed specifically for the measurement and analysis of web page download times.
- YSlow 2.0.7 – the Mozilla Firefox add-on that analyses web page download times.
- Apache 2.2.11 – an Open Source web server.
- CakePHP 1.2.5 – an Open Source PHP rapid development framework.
- PHP 5.3.0 – an Open Source web scripting language.
- MySQL 5.1.36 – an Open Source database management system (DBMS).

All the software packages listed above were chosen because of their open nature, which permitted the researcher to use them without first acquiring a license. Another reason for using these open source products was because of their popularity with web developers, as each accounts for more than half of the market share in its respective area.

Hardware

The measurements in this research were all taken on one machine which acted as both the server and the client. The machine had the following specifications:

- Microsoft Windows XP Professiional Service Pack 2 Operating System
- Pentium(R) Dual-Core CPU E5200 @ 2.50GHz
- 2.49GHz, 0.99GB of RAM

V. DATA COLLECTION: RESPONSE TIME MEASURING TECHNIQUES

The following methods for getting an accurate measurement of a web page's response time were considered:

- Page with frames, and
- Using YSlow.

A quick review of the two methods follows below.

A. *Using YSlow*

YSlow is a Mozilla Firefox add-on component that is designed to help web developers analyse the performance of their web pages. YSlow uses JavaScript to analyse web page performance by examining all the components on the page, including dynamically created components. Though YSlow provides a lot of information about where performance could be lacking, such information is not necessary for the purpose of this research and will thus not be considered. The only piece of information that is required is the time it takes for a particular web page to load. A page viewer can access YSlow at the bottom right-hand corner of the Mozilla Firefox browser.

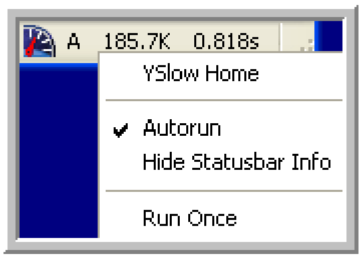


Figure 5.1: Appearance of YSlow

The far right section of YSlow (with value 0.818s in Figure 5.1) shows the load time for the web page, while the 185.7K shows the size of the page and its embedded components. By checking Autorun as shown in Figure 5.1, YSlow would run each time a page is loaded in the browser. If many tabs are opened, then the values shown would be for the page in the tab which was loaded last.

Advantages of this method

• No need for adding extra JavaScript code to the individual pages. Once YSlow is installed in the browser, then any page can be analysed without any modification to the actual page. YSlow also requires no configuration in order to work in the browser.
• Little or no overhead experienced due to the use of YSlow. Since it is a browser plug-in, it does not affect the page load time as it is independent of the page.
• YSlow is an internationally tested tool which already has a reputation of accurately measuring response time.

Disadvantages

• At the time of this research, YSlow was only supported by Mozilla Firefox, thus the tool could not be used to measure response times in other browsers. It would have been more credible to measure response times of a single page in many different browsers so as to eliminate the choice of browser as a limiting factor.

After considering the advantages and disadvantages of each of the two methods mentioned above, as well as taking into consideration the project timeframe, YSlow was the natural choice for the purpose of this research. Thus YSlow was used for all the measurements stated in this study and consequently Mozilla Firefox was the only browser used for viewing the pages.

VI. RESEARCH DESIGNS: MEASUREMENT OF RESPONSE TIME

Three identical dynamic web pages were created for testing in this experiment. The first page was created using pure PHP code, while the second was created with the CakePHP framework. The third page was created using CakePHP with the proposed technique of using template-loading. The three pages were then loaded alternatively in the browser for twenty times, at different times of the day. Thus, in total there were sixty page loads considered for this test. The average response time was then calculated for each of the three pages.

The aim of the first page was to come up with a rough estimate of the overhead in fulfilment of the first objective. The first page also acts as a benchmark against which the performance of the third page will be evaluated.

A. *The new approach*

The proposed technique was then implemented on the CakePHP page. This technique proposes the use of AJAX technology to load dynamic web pages as quasi static templates. All the static components of a web page are loaded first before any dynamic variables are displayed. With this approach, a user perceives the page to be fully loaded as soon as the layout appears on screen. However the actual requested content would not have loaded at this time, but the user may nevertheless feel more comfortable when seeing the partial display. The user may also start to browse over the partial browser display while waiting for the rest of the page to display. This is in accordance with the definition b) of response time as given in Section 2.3.

In order for this technique to be possible, two main properties of web pages were used:
• The JavaScript onload event handler, and
• The XMLHttpRequest object of AJAX

The onload event handler is used to trigger the loading of dynamic content after the static content had finished loading. This can be achieved in one of two ways as described below:

- In the body tag of the HTML file put <body onload= “process(myElement, targetDiv)”>, or
- In the JavaScript file put the line window.onload= process(myElement, targetDiv);

The process function creates an XMLHttpRequest object and sends a request for the element myElement from the server. When returned, myElement is added to the div with id targetDiv. The process of updating the page is handled in the handleRequestStateChange function.

```
function process(myElement, targetDiv){
      var xmlHttp = createXmlHttpRequestObject();
      // only continue if xmlHttp isn't void
      if (xmlHttp){
            // try to connect to the server
            try{
                  // initiate reading the a file from the server
                  xmlHttp.open("GET",
            "http://localhost/myElement/loadElement.php?myfile="+myElement,
            true);
            xmlHttp.onreadystatechange = handleRequestStateChange;
            xmlHttp.send(null);
               }
               // display the error in case of failure
               catch (e){
                  alert("Can't connect to server:\n" + e.toString());
               }
         }
```

Listing 1: The process function

From Listing 1, whenever the readyState attribute of the XMLHttpRequest object changes, the handleRequestStateChamge function is called. The readyState changes value from 0 to 4 depending on the status of the request. The meaning of each state is given below:
0 = uninitialized;
1 = loading;
2 = loaded;
3 = interactive;
4 = complete.

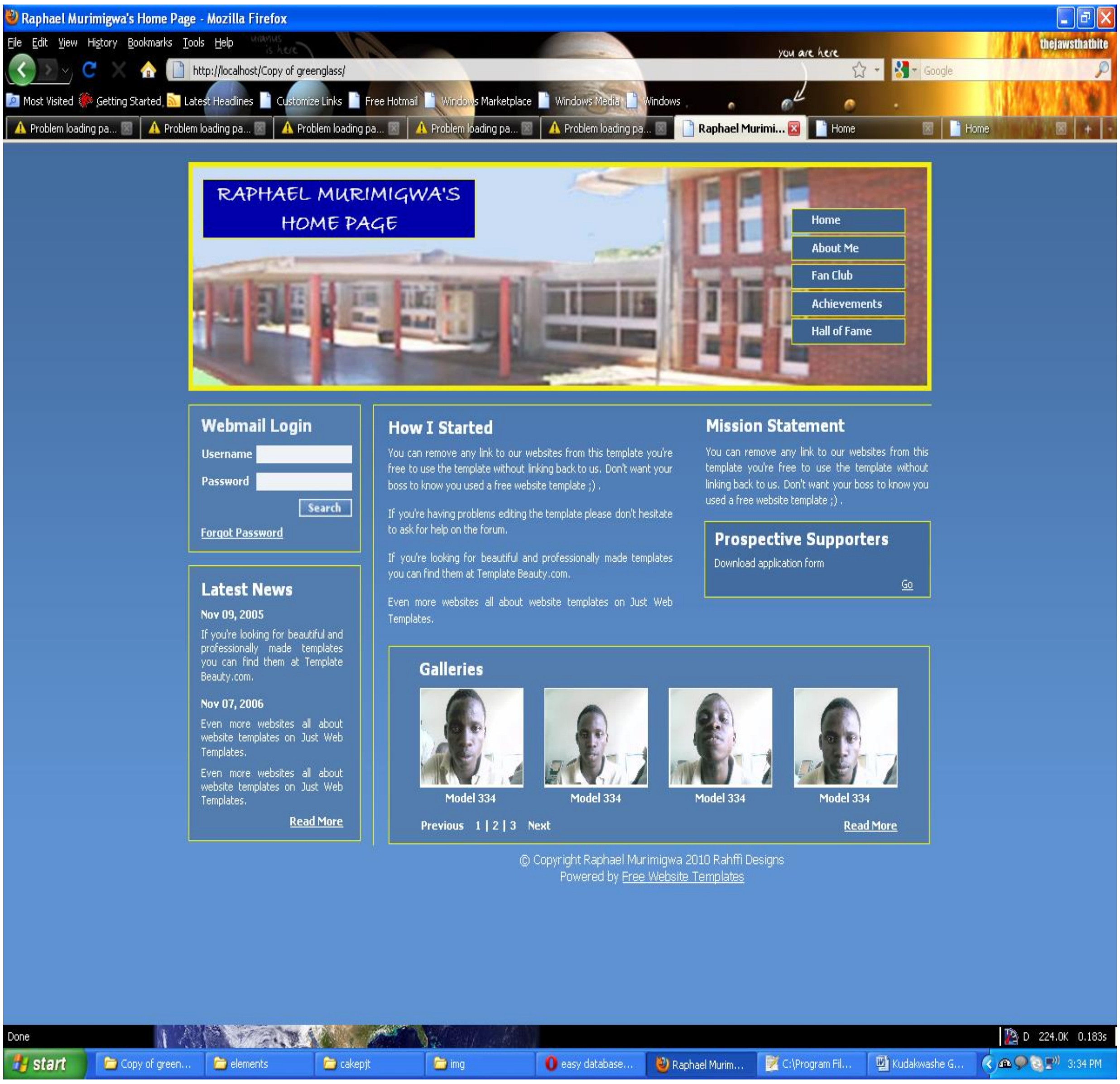
```
function handleRequestStateChange(){
        // obtain a reference to the <div> element on the page
        myDiv = document.getElementById(targetDiv);
        // when readyState is 4, we read the server response
        if (xmlHttp.readyState == 4){
                // continue only if HTTP status is "OK"
                if (xmlHttp.status == 200){
                        try{
                                // read the message from the server
                                response = xmlHttp.responseText;
                                // display the message
                                myDiv.innerHTML += response;
                        }
                        catch(e){
                                …
                        }
                }
                else{
                        …
                }
        }
}
```

Listing 2: The handleRequestChange function

The handleRequestChange function gets the value of targetDiv from the process function. It then uses the DOM functions to reference targetDiv. With this reference, the response from myElement is added to the specified div using the DOM innerHTML function. This updating of the page is repeated for every div element that needs to be updated with dynamic content on the page. The calls to the process function can be done asynchronously and concurrently, thus greatly increasing the page load speed.

VII. POPULATION AND SAMPLE

The sample used in this research was a collection of three web pages. All three pages had an identical interface, but the technology used to build the pages was different. The page layout of all the three pages was as shown in Figure 7.1 below.



Webmail Login

How I Started

You can remove any link to our websites from this template you're free to use the template without linking back to us. Don't want your boss to know you used a free website template ;) .

If you're having problems editing the template please don't hesitate to ask for help on the forum.

If you're looking for beautiful and professionally made templates you can find them at Template Beauty.com.

Even more websites all about website templates on Just Web Templates.

Mission Statement

You can remove any link to our websites from this template you're free to use the template without linking back to us. Don't want your boss to know you used a free website template ;) .

Prospective Supporters

Download application form

Latest News

Nov 09, 2005

If you're looking for beautiful and professionally made templates you can find them at Template Beauty.com.

Nov 07, 2006

Even more websites all about website templates on Just Web Templates.

Even more websites all about website templates on Just Web Templates.

Galleries

© Copyright Raphael Murimigwa 2010 Rahffi Designs
Powered by Free Website Templates

Figure 7.1: The layout of all the three pages used in the research

This design was of a personal homepage for the researcher. The purpose of having used a home page, and not an inner page was because the proposed technique is best suited for home pages as web users usually enter a site through the home page. The same images and script files (JavaScript and CSS) were used for all three pages. The reason for doing this was to eliminate file size as a bottleneck in response time. However, because of the different technologies used in building the web pages, there was expected to be some difference in the responsiveness of the pages.

VIII. Data Analysis Procedure

The method used for analysing the results of this study was to compute the average response time for each page. The highest and lowest response times were also taken note of. The goal was then to compare these values for each of the three pages, in order to rank the pages according to responsiveness.

IX. Analysis and interpretation of results

Response time PHP vs Cake PHP

This section presents the results obtained after measure the overhead in response time of CakePHP pages against those for PHP.

Table 9.1: Raw results from measurements

Page Load Number	**Response Time (seconds)**	
	PHP page	**CakePHP page**
1	0.173	0.252
2	0.189	0.246
3	0.143	0.271
4	0.183	0.289
5	0.206	0.264
6	0.174	0.262
7	0.180	0.253
8	0.217	0.249
9	0.175	0.258
10	0.170	0.249
11	0.178	0.263
12	0.177	0.248
13	0.180	0.237
14	0.215	0.265
15	0.177	0.244
16	0.165	0.250
17	0.190	0.278
18	0.184	0.276
19	0.174	0.241
20	0.178	0.261

Table 9.1 shows the response time measurements from the two pages created using PHP and CakePHP. The information in Table 9.1 is reproduced in Figure 9.1 in the form of a bar graph.

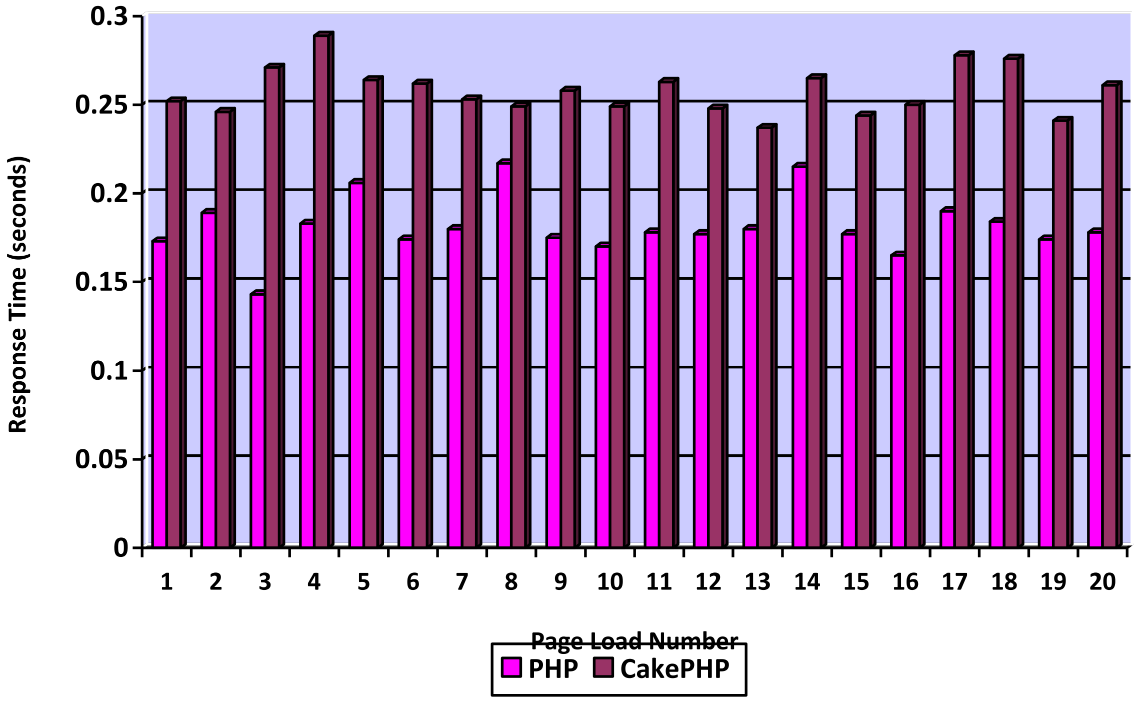


Figure 9.2: Graphical comparison of PHP and CakePHP response times

The data in Table 9.1 and Figure 9.2 is not complete for analysis, thus in order for an analysis of the data to be made, additional information is required on the distribution of the response times.

Table 9.2: Statistical information about data in Table 9.1

	PHP	**CakePHP**
Total Response time (seconds)	3.628	5.156
Mean (Total/20)	0.1814	0.2578
Minimum	0.143	0.237
Median	0.178	0.2555
Maximum	0.217	0.289
Standard deviation	0.016706	0.013466
Variance	0.000279	0.000181

Table 9.2 provides additional statistical information about the values presented in Table 9.1.Of the information presented in Table 9.2, the mean, minimum, median, and maximum are suitable for direct comparison as in Figure 9.3.

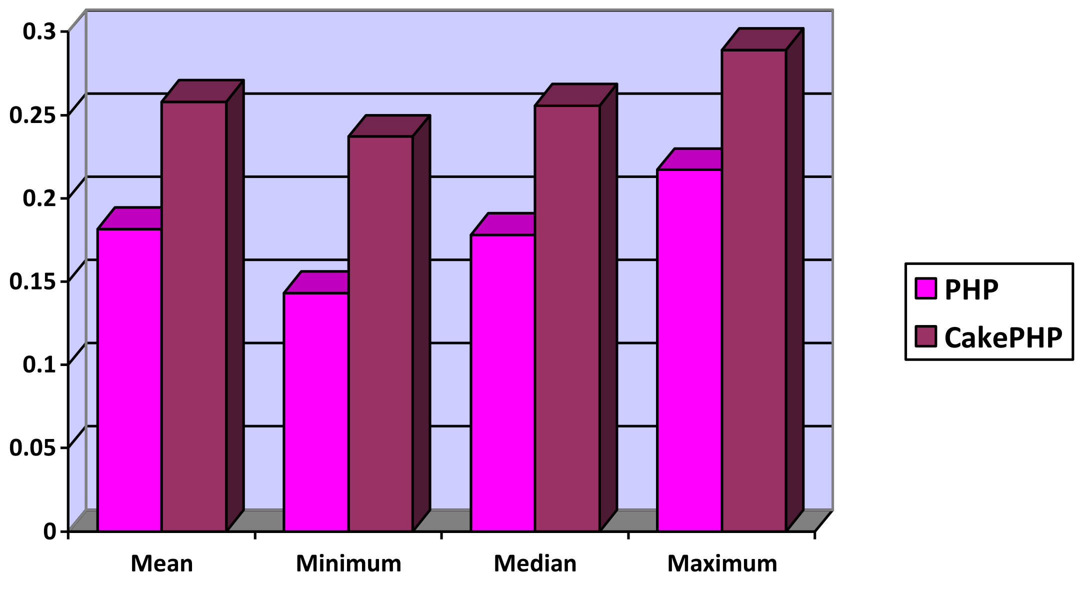


Figure 9.3: Comparison of response time distributions

The graph above shows the mean response time for the two pages. From the graph, it can be seen that CakePHP takes longer to respond on average than PHP, as it has higher values for each point of comparison. The next task is to calculate a numerical value for the overhead. This is done using the formula below.

$$\frac{\text{CakePHP response time}}{\text{PHP response time}} \times 100\% = \text{Overhead as a percentage}$$

Equation 1: Formula for CakePHP overhead

By using this formula with the values for mean in Table 9.2, the result obtained is 142.12%. This value shows that the CakePHP page requires an extra 42% of the time needed by the PHP page to fully load in the browser.

Response time Cake PHP vs Improved Cake PHP

In light of the weaknesses of CakePHP, as exposed by the results in the previous section, the next step was to implement a page-loading technique that would reduce the user-perceived response time of web pages and then ascertain the effectiveness of the proposed technique. The results presented in this section show the response time of the CakePHP page after having implemented the loading technique. For simplicity, the page that utilises the implemented technique is simply referred to as the improved CakePHP page. Table 9.3 shows the twenty response time measurements for the improved CakePHP page.

Table 9.3: Response time measurements for improved CakePHP page

Page Load Number	Response time (seconds)
1	0.185
2	0.185
3	0.188
4	0.198
5	0.191
6	0.186
7	0.192
8	0.192
9	0.189
10	0.200
11	0.207
12	0.210

13	0.187
14	0.200
15	0.203
16	0.188
17	0.197
18	0.185
19	0.197
20	0.188

In order to analyse the measurements in Table 9.3 effectively against those presented in Table 9.1, a more visual approach has to be taken. Figure 9.4 is a direct comparison of the two CakePHP pages. From the graph, the proposed technique can be seen to make a significant improvement in the response time.

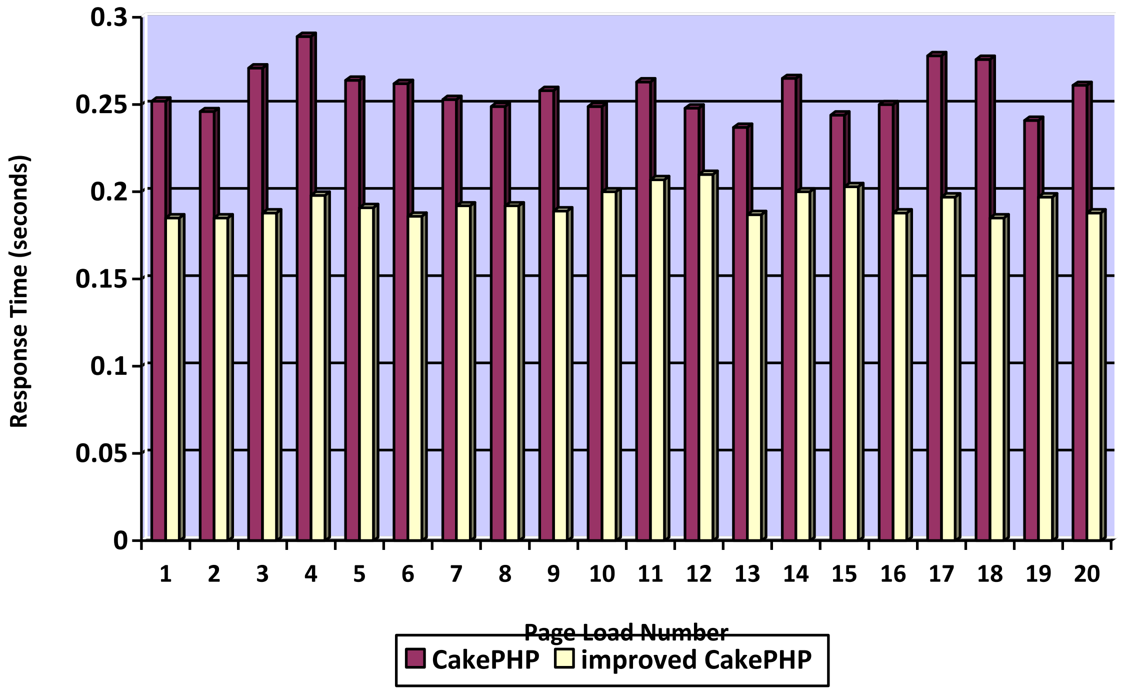


Figure 9.4: Graphical comparison of response times for CakePHP and improved CakePHP

Table 9.4 shows the statistical distribution of the data in Table 9.3

Table 9.4: Statistical distribution of response time for improved CakePHP page

Total Response time (seconds)	3.868
Mean (Total/20)	0.1934
Minimum	0.185
Median	0.1915
Maximum	0.21
Standard deviation	0.007646
Variance	0.000181

Figure 9.5 is a statistical comparison of the data presented in Figure 9.4. It is an analysis of the distribution of the response time measurements for the two CakePHP pages.

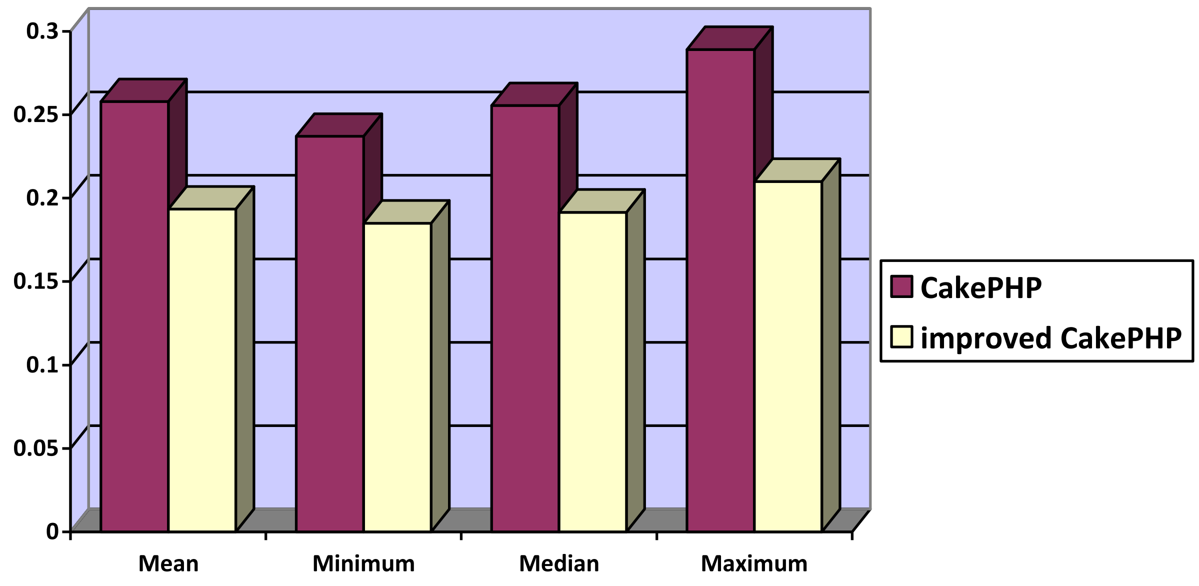


Figure 9.5: Statistical comparison of the CakePHP and improved CakePHP pages

The mean loading times can now be used to calculate the overhead of the improved CakePHP page over the PHP page using a formula similar to the one in Figure 9.3. In this case, the mean CakePHP response time is replaced by the improved CakePHP page's mean response time. This calculation produces a value of 106.62%, showing an overhead of only 6.62%. This value represents a significant improvement from the normal CakePHP page which has an overhead of 42.12%. The improvement results in a cut of 35.5% in response time. This information is represented in Table 9.5.

Table 9.5: Response time overhead

	CakePHP	Improved CakePHP	Difference
Overhead (%)	42.12	6.62	35.5

By looking back at Table 9.2 and Table 9.4, it can be seen that the variance of all the sets of data presented here is very small, showing that the data is consistent and in a defined range. Figure 9.6 is a combination of Figure 9.1 and Figure 9.4, and it compares all the three pages used in this research.

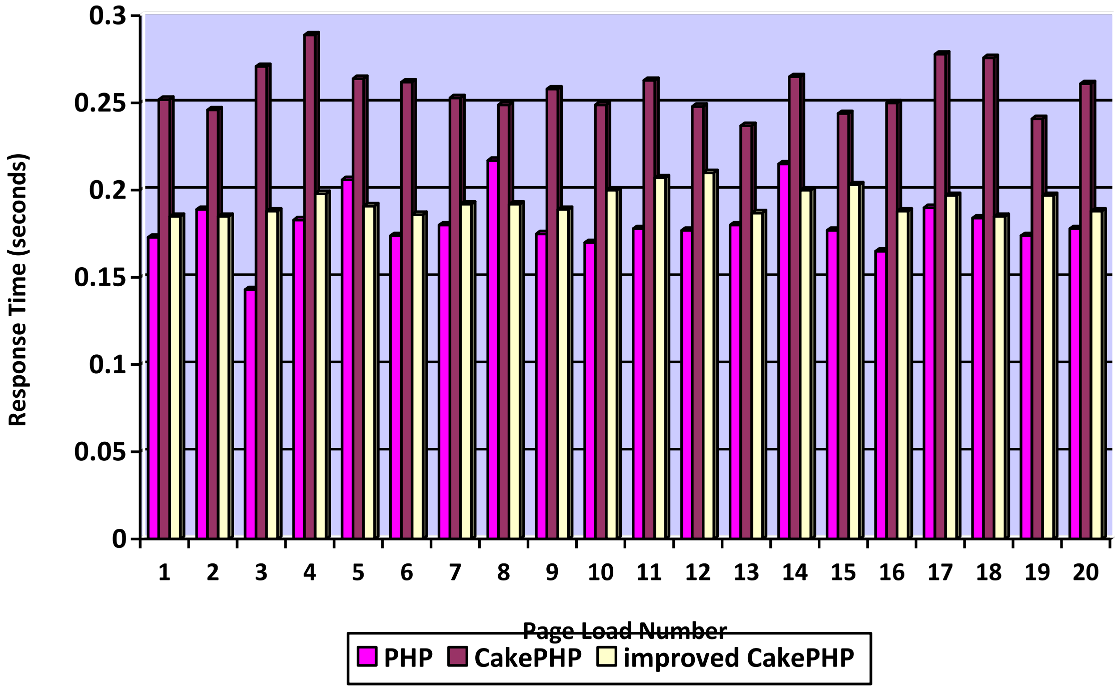


Figure 9.6: Comparison of all three page response time measurements

Further, Figure 9.7 is a combination of Figure 9.2 and Figure 9.5, and it presents a comparison of the statistical distribution of the three pages used in the research.

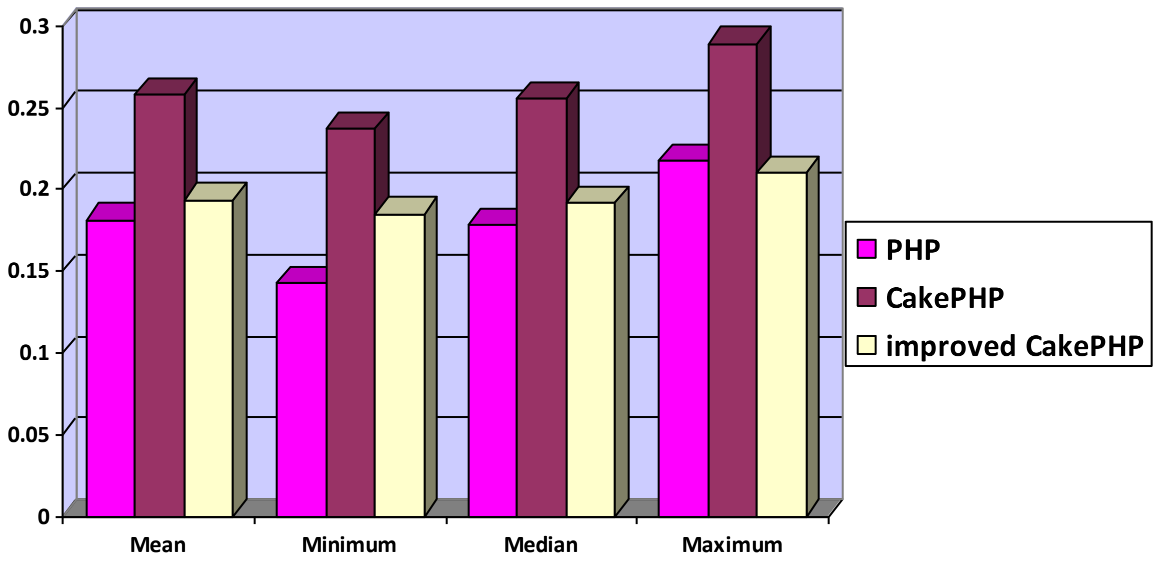


Figure 9.7: Statistical comparison of the three pages

With the two figures above, a final analysis of the research findings can now be made.

X. SUMMARY OF RESEARCH FINDINGS

The conclusion on the findings of this research is as presented in Figure 9.6 and Figure 9.7. The PHP page has the best performance of all the three pages, followed by the improved CakePHP page. The CakePHP page performs worst of all the pages because of the translation overhead involved. Even though the PHP page performs marginally better than the improved CakePHP page, sometimes it is wiser to use the latter because of the very short development time involved. This is true particularly for large projects which have to be delivered in a very short space of time. Development teams will therefore have to make a compromise between development time and response time and select the technique best suited to the project at hand. It was also noted that implementing the improved CakePHP technique proposed by this research would improve the end user perceived CakePHP response time by as much as 35%. This can be a significant improvement, particularly for web sites which handle a lot of requests at any given time.

REFERENCES

[1] Broadwell, P. M. 2004. Response Time as a Performability Metric for Online Services. Report No. UCB/CSD-04-1324: University of California, Berkeley.

[2] Cecchet, E., Chanda, A., Elnikety, S., Marguerite, J., & Zwaenepoel, W. 2003. Performance Comparison of Middleware Architectures for Generating Dynamic Web Content. In the 4th ACM/IFIP/USENIX International Middleware Conference, Rio de Janeiro, 16-20, June 2003.

[3] Chandranmenon, G. P., & Varghese, G. 2001. Reducing Web Latency Using Reference Point Caching. In the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001), Anchorage, 22-26 April 2001.

[4] Cherkasova, L., Fu, Y., Tang, W., & Vahdat, A. 2003. Measuring and Characterizing End-to-End Internet Service Performance. ACM Transactions on Internet Technology, 3(4): 347-391.

[5] Choo, H.C. & Lee, S.P., 2008. Towards persistence framework-based Rapid Application Development toolkit for Web Application Development. Journal of Computer Science 4 (4): 290-297.

[6] Davidson, N. & Galbraith, S. Quantifying the relationship between website download time and abandonment by users. [Online]. Available: http://www.red-gate.de/products/ants_load/technical_papers/understanding_testing_results.htm [25 April 2010].

[7] Diao, Y., Hellerstein, J. L., Parekh, S., & Bigus, J. P., 2003. Managing Web Server Performance with AutoTune Agents. IBM Systems Journal, 42(1): 136-149.

[8] Garrett, J.J., 2010. Ajax: A New Approach to Web Applications. [Online]. Available: http://www.adaptivepath.com/ideas/essays/archives/000385.php [23 April 2010]

[9] Haddad, I., 2007 Adopting an Open Source Approach to Software Development, Distribution, and Licensing. [Online]. Available: http://opensource.sys - con.com/read/318776.htm [23 March 2010].

[10] Hitz, M., Leitner, G., & Melcher, R., 2006. Usability of web applications. In Kappel, G., Proll, B., Reich, S. & Retschitzegger, W. (Eds.). Web Engineering, pp. 219-246. Heidelberg: John Wiley & Sons.

[11] Hoxmeier, J. A., & Dicesare, C., 2000. System Response Time and User Satisfaction: An Experimental Study of Browser-based Applications. In Proceedings of the 4th CollECTeR Conference on Electronic Commerce, Breckenridge 11 April 2000.

[12] Iyengar, A., Nahum, E., Shaikh, A., & Tewari, R. 2002. Enhancing Web Performance. In the 2002 IFIP World Computer Congress (Communication Systems: The State of the Art), Montreal, 25-30 August 2002.

[13] Kappel, G., Proll, B., Reich, S. & Retschitzegger, W. 2006 Web Engineering, Glasgow, John Wiley & Sons Ltd.

[14] Khan, A. 2004. Survey of development methodologies for the web. Department of Information Systems. The University of Melbourne.

[15] Khosrowpour, M. & Herman, N. 2000. "Web-Enabled Technologies Assessment and Management: Critical Issues, Challenges and Trends," In Managing Web-Enabled Technologies in Organizations: A Global Perspective, M. Khosrowpour (Ed.), Idea Group Publishing, Hershey, PA, 2000, pp. 1–22.

[16] Killelea, P. 2002. Web Performance Tuning. Sebastopol: O'Reilly & Associates.

[17] Kohavi, R. 2001. Mining e-Commerce Data: The Good, the Bad, and the Ugly. In the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, 26-29 August 2001.

[18] Kotsis, G. 2006. Performance of Web Applications. In Kappel, G., Proll, B., Reich, S. and Retschitzegger, W. (Eds.). Web Engineering, pp. 247-264. Heidelberg: John Wiley & Sons.

[19] Krishnamurthy, B., & Wills, C. E. 2000. Analyzing Factors that Influence End-to-End Web Performance. Computer Networks, 33: 17-32.

[20] Krishnamurthy, B., & Wills, C. E. 2002. Improving Web Performance by Client Characterization Driven Server Adaptation. In Proceedings of the 11th International Conference on World Wide Web, Honolulu, 7-11 May 2002.

[21] Marshak, M., & Levy, H. 2003. Evaluating Web User Perceived Latency Using Server Side Measurements. Computer Communications, 26(8): 872-887.

[22] Martin J. 1990. RAD, Rapid Application Development. MacMillan Publishing Co., New York.

[23] McDonald, A. & Welland, R. 2001a. Web Engineering in Practice. 10th international conference on WWW. Hong Kong.

[24] McDonald, A. & Welland, R. 2001b. Agile Web Engineering (AWE) Process University of Glasgow 2001, Department of Computing Science Technical Report TR-2001-98

[25] Nah, F. 2003. A Study on Tolerable Waiting Time: How Long are Web Users Willing to Wait? In American Conference on Information Systems (AMCIS), Tampa, 4-6 August 2003.

[26] Nielsen, J. 1997. The Need for Speed. Available: http://www.useit.com/alertbox/9703a.html [8 February 2010].

[27] Nielsen, J. 1999. User Interface Directions for the Web. Communications of the ACM, 42(1): 65-72.

[28] Nielsen, J. 2000. Designing Web Usability: The Practice of Simplicity. Indianapolis: New Riders Publishing.

[29] Rajamony, R., & Elnozahy, M. 2001. Measuring Client-Perceived Response Times on the WWW. In the 3rd USENIX Symposium on Internet Technologies and Systems (USITS), San Francisco, 26-28 March 2001.

[30] Rose, G., Khoo, H. & Straub, D. 1999. "Current Technological Impediments to Business-to-Consumer Electronic Commerce," Communications of the AIS (1:16), 1999, pp. 1–74.

[31] Rosenstein, M. 2000. What is Actually Taking Place on Web Sites: e-Commerce Lessons from Web Server Log. In the 2nd ACM Conference on Electronic Commerce, Minneapolis, 17-20 October 2000.

[32] Steinberg, J., & Pasquale, J. 2002. A Web Middleware Architecture for Dynamic Customization of Content for Wireless Clients. In the 11th International World Wide Web Conference (WWW2002), Honolulu, 7-11 May 2002.