

AALTO UNIVERSITY SCHOOL OF ELECTRICAL ENGINEERING

Department of Communications and Networking

Shuang Gu

Page Load Performance Evaluation for Mobile Browser on a Cloud  
Computing Platform

Master's Thesis submitted for the degree of Master of Science in Technology.

Espoo, 26 May, 2011

Supervisor: Adjunct Professor Timo O. Korhonen

Instructor: Adjunct Professor Timo O. Korhonen

**Author:** Shuang Gu

**Name of the thesis:** Page Load Performance Evaluation for Mobile Browser on a Cloud Computing Platform

**Date:** 26 May, 2011

**Number of pages:** 12 + 81

**Faculty:** Faculty of Electronics, Communications and Automation

**Professorship:** S-72 Communications

**Supervisor:** Adjunct Professor Timo O. Korhonen

**Instructor:** Adjunct Professor Timo O. Korhonen

**Abstract text:**

More and more network data flows of the Internet are nowadays contributed by wireless access which is initiated from mobile devices, such as smartphones and tablets. As a result, the performance of mobile browser applications integrated in software stacks of these portable devices draws more and more attentions. In order to meet the needs of software engineering and benchmarking among various device vendors particularly, it is getting increasingly important to evaluate the performance of *mobile browser* accurately and efficiently. However, due to the distinct hardware and software characteristics, conventional measurement methods which are widely deployed in desktop browsing are not so applicable for mobile browsers.

This study tries to solve this problem by finding out a practical approach to evaluate the *page load performance* of mobile browsers without additional software and hardware measurement investments. Several measurement models are therefore introduced, discussed and evaluated in order to point out the accurate and efficient one. As a benchmarking tool, an online measurement application is designed and implemented in the experimental part of this study. Verification data are then analyzed and discussed.

The implementation of such a web application and its context of the evaluation model provide an outline of an ideal approach to benchmark page load performance of mobile browsers. This optimized approach not only reduces the complexity of configuring a measurement environment for mobile devices, but also provides relatively accurate measurement results for continuous evaluation of mobile browsers' *page loading speed*, which is useful for developers to increase mobile browser's performance incrementally. The collected measurement results using such a measurement model are also useful as a reference for benchmarking various mobile browsers from different manufacturers.

Our research found out also possible factors which may influence to the performance of mobile browsers. The practice targeting of these factors, such as the design of web pages, will help users of mobile devices to achieve better user experience when surfing the Internet.

**Keywords:**

Mobile browser, Page load performance, Page loading speed, Cloud computing, JavaScript, Ajax, Google App Engine

## Preface

I started to be interested in investigating on an easier method to measure the page loading speed of mobile browsers since one and half years ago when I was working as a consulting software tester in Nokia's Maemo browser team which was working on a customized version of Mozilla Firefox Mobile. The efforts on the development of ideas and practical codes and finally summarized into this literature as my Master's thesis in Helsinki University of Technology.

I need to thank my current employer, HiQ, on providing me an opportunity to work together with the Maemo Browser team in Nokia, so that I can get familiar with the concepts of performance evaluation of mobile browsers to generate the idea of this study. And thanks to Google to provide its App engine platform, without its ease to use, I would not be able to finish coding and deployment of such a kind of coding work.

I want to thank my supervisor Timo O. Korhonen, who helped me a lot for turning my code into this study paper, and Martin William, who helped me with the language checking of the final thesis manuscript.

I would like to thank my parents who are living in my home city in China now. Without their support, both financially and spiritually, I would not have managed to complete this study.

Special thanks to my husband, Hengzhi Liu, for always believing in and encouraging me, and for having the patience to take care of our baby daughter when I had to steal time to work on completing this Master's thesis.

And also, best wishes to Cindy, my beautiful angle.

Espoo, 26 May, 2011

Shuang Gu

## Table of Contents

|   |     |
|---|-----|
| Preface .....                                       | ii  |
| Table of Contents .....                             | iii |
| List of Figures .....                               | vi  |
| List of Abbreviations .....                         | vii |
| List of Concepts .....                              | ix  |
| 1. Introduction .....                               | 13  |
| 1.1. Purpose of the Study.....                      | 15  |
| 1.2. Research Questions .....                       | 17  |
| 1.3. Structure of the Thesis.....                   | 19  |
| 2. Literature Overview .....                        | 22  |
| 2.1. Mobile Browser.....                            | 22  |
| 2.1.1. Definition of Mobile Browser.....            | 22  |
| 2.1.2. Structure of Mobile Browser .....            | 23  |
| 2.1.3. Performance Metrics of Mobile Browser .....  | 25  |
| 2.2. Page Load Performance .....                    | 27  |
| 2.2.1. Definition of page load performance.....     | 28  |
| 2.2.2. Measurement of page load performance.....    | 28  |
| 2.3. Utilities for Web Application Development..... | 29  |
| 2.3.1. Ajax.....                                    | 29  |
| 2.3.2. JavaScript library .....                     | 30  |
| 2.3.3. Web application framework .....              | 31  |
| 2.4. Cloud Computing .....                          | 32  |
| 2.4.1. Definition of cloud computing .....          | 32  |
| 2.4.2. Google App Engine .....                      | 33  |
| 3. Research.....                                    | 34  |
| 3.1. Research Method.....                           | 34  |
| 3.2. Requirement Analysis .....                     | 36  |
| 3.2.1. Functionality .....                          | 37  |
| 3.2.2. Portability.....                             | 37  |

|  |    |
|--|----|
| 3.2.3. Cost restriction.....                       | 38 |
| 3.2.4. Manual vs. automatic.....                   | 38 |
| 3.3. Design.....                                   | 39 |
| 3.3.1. Measurement modules.....                    | 39 |
| 3.3.2. User interface of measurement control ..... | 41 |
| 3.3.3. Possible measurement models .....           | 43 |
| 3.4. Implementation .....                          | 48 |
| 3.4.1. Choice of hosting platform .....            | 48 |
| 3.4.2. Selection of development tools .....        | 49 |
| 3.4.3. Capacity of measurement model.....          | 50 |
| 3.5. Verification.....                             | 50 |
| 3.5.1. Browser compatibility .....                 | 51 |
| 3.5.2. Category of mirrored web pages .....        | 51 |
| 3.6. Deployment and Maintenance .....              | 53 |
| 3.6.1. Deployment.....                             | 53 |
| 3.6.2. Maintenance .....                           | 53 |
| 4. Results.....                                    | 54 |
| 4.1. Decision of Design.....                       | 54 |
| 4.1.1. Measurement model .....                     | 54 |
| 4.1.2. Measurement process .....                   | 55 |
| 4.2. Measurement practices .....                   | 58 |
| 4.2.1. Measurement participants .....              | 58 |
| 4.2.2. Verification across browsers.....           | 59 |
| 4.2.3. Verification across web pages.....          | 60 |
| 5. Discussion.....                                 | 62 |
| 5.1. Answering Research Questions.....             | 62 |
| 5.2. Benchmarks.....                               | 66 |
| 5.2.1. Comparison with SunSpider.....              | 66 |
| 5.2.2. Comparison with Speed-battle .....          | 67 |
| 5.2.3. Comparison with Numion Stopwatch.....       | 67 |

|   |    |
|---|----|
| 6. Conclusion.....                                    | 69 |
| 6.1. SWOT Analysis.....                               | 69 |
| 6.2. Challenge of the study .....                     | 75 |
| 6.3. Possible future work .....                       | 76 |
| Bibliography .....                                    | 77 |
| Appendices.....                                       | 81 |
| Appendix A: Source Code of the Experimental Work..... | 81 |

## List of Figures

|  |    |
|--|----|
| <b>Figure 1-1:</b> The infrastructure of mobile access to the Internet (Ye, 2010) .....        | 14 |
| <b>Figure 1-2:</b> Scope of the study .....  | 20 |
| <b>Figure 2-1:</b> High level structure of web browsers .....                                  | 24 |
| <b>Figure 2-2:</b> Cloud computing visual diagram .....  | 32 |
| <b>Figure 3-1:</b> The Waterfall model of software engineering process .....                   | 35 |
| <b>Figure 3-2:</b> The agile model of software engineering process applied in this study ..... | 35 |
| <b>Figure 3-3:</b> The Projects Management Triangle .....                                      | 38 |
| <b>Figure 3-4:</b> Modular components of automatic measurement environment.....                | 40 |
| <b>Figure 3-5:</b> UI draft on view of new execution .....                                     | 42 |
| <b>Figure 3-6:</b> UI draft on view of running execution.....                                  | 42 |
| <b>Figure 3-7:</b> UI draft on view of execution results .....                                 | 43 |
| <b>Figure 3-8:</b> A measurement model, web page serving on device .....                       | 44 |
| <b>Figure 3-9:</b> A measurement model, web page serving in local environment.....             | 45 |
| <b>Figure 3-10:</b> A measurement model, web page serving distributed .....                    | 46 |
| <b>Figure 3-11:</b> A measurement model, web page serving in cloud .....                       | 47 |
| <b>Figure 3-12:</b> Top 35 of Global Top Sites retrieved from Alexa.com .....                  | 52 |
| <b>Figure 4-1:</b> Screenshot of Main page, captured by using Nokia N900 .....                 | 56 |
| <b>Figure 4-2:</b> Screenshot of Run page, captured by using Nokia N900 .....                  | 57 |
| <b>Figure 4-3:</b> Screenshot of Result page, captured by using Nokia N900.....                | 57 |
| <b>Figure 4-4:</b> Mobile browsers within devices in measurement .....                         | 58 |
| <b>Figure 4-5:</b> Page loading speed across browsers.....                                     | 59 |
| <b>Figure 4-6:</b> Average page loading speed.....   | 61 |
| <b>Figure 6-1:</b> The SWOT model (CIPD, 2008) .....   | 69 |

## List of Abbreviations

|      |                                   |
|------|-----------------------------------|
| Ajax | Asynchronous JavaScript and XML   |
| API  | Application Programming Interface |
| CSS  | Cascading Style Sheets            |
| CPU  | Central Processing Unit           |
| CRUD | Create, Read, Update, and Delete  |
| DOM  | Document Object Model             |
| DRAM | Dynamic Random-Access Memory      |
| DRY  | Don't Repeat Yourself             |
| EC2  | Elastic Compute Cloud             |
| FTP  | File Transfer Protocol            |
| GUI  | Graphical User Interface          |
| HCI  | Human-Computer Interaction        |
| HTML | Hypertext Markup Language         |
| HTTP | Hypertext Transfer Protocol       |
| IIS  | Internet Information Services     |
| IO   | Input and Output                  |
| JSON | JavaScript Object Notation        |
| JVM  | Java Virtual Machine              |
| LAMP | Linux, Apache, MySQL, and PHP     |
| MTV  | Model-Template-View               |
| MVC  | Model-View-Controller             |



|       |   |
|-------|---|
| ORM   | Object-Relational Mapping                         |
| OS    | Operating System                                  |
| REST  | Representational State Transfer                   |
| RIA   | Rich Internet Application                         |
| RSS   | Really Simple Syndication / Rich Site Summary     |
| SDK   | Software Development Kit                          |
| SQA   | Software Quality Assurance                        |
| SQL   | Structured Query Language                         |
| SWOT  | Strengths, Weaknesses, Opportunities, and Threats |
| UI    | User Interface                                    |
| URI   | Uniform Resource Identifier                       |
| URL   | Uniform Resource Locator                          |
| USB   | Universal Serial Bus                              |
| UX    | User Experience                                   |
| WLAN  | Wireless Local Area Network                       |
| YAML  | YAML Ain't Markup Language                        |
| XHTML | Extensible HyperText Markup Language              |
| XML   | Extensible Markup Language                        |

## List of Concepts

|                               |  |
|-------------------------------|--|
| Ajax                          | refers to a group of interrelated web development techniques used on the client-side to create interactive web applications. [Section 2.3.1]   |
| Cloud computing               | refers to Internet-based development and use of computer technology. Instead of traditional client-server structure, cloud computing abstracts the details of control over technology infrastructure from users and sets up a new supplement, consumption, and delivery model based on the Internet. It typically involves the provision of dynamically scalable and often virtualized resources as a service over the Internet. [Section 2.4.1] |
| Convention over configuration | refers to a software design philosophy and technique with a strategy of defaults over explicit configuration. (Miller, 2009)   |
| DOM events                    | refer to Document Object Model (DOM) events which allow JavaScript language to listen and handle inside a HTML document being manipulated by a web browser.  |
| Don't Repeat Yourself         | refers to a principle of software development to reduce repetition of source code and all related information of software engineering. (Hunt, 2010)  |
| Facebook                      | refers to the most popular social networking application. It enables e.g. creating your own profile, contacting friends, and sharing photos. (Facebook, 2010)  |
| Google App Engine             | refers to a development environment for web applications and a hosting service provided by Google. It consists of a Python-based web framework and an un-relational data base, which is called Bigtable. For personal usage, it is free of charge to register and use with CPU and storage limitations. [Section 2.4.2]  |
| JavaScript library            | refers to a library of pre-written JavaScript controls which provides convenience to development of JavaScript-based   |

applications, especially for Ajax and other web-centric technologies. [Section 2.3.2]

|                           |  |
|---------------------------|--|
| jQuery                    | refers to a fast and concise JavaScript library that simplifies HTML document traversing, event handling, animating, and Ajax interactions for rapid web development. [Section 2.3.2] jQuery is designed to change the way that you write JavaScript. (jQuery, 2006) |
| Language runtime          | refers to one of the standard web infrastructures on server side which explains and executes computer language scripts or binary executables. This includes, for instance, PHP's runtime, Python's, Ruby's, JVM (Java Virtual Machine), and so on.                   |
| Mobile browser            | refers to a web browsing software application designed for use on various mobile devices, such as smartphones or tablet computers. [Section 2.1.1]   |
| Model-View-Controller     | refers to a software architectural pattern widely accepted in structural design of web application frameworks.   |
| Object-relational mapping | refers to a programming technique for converting data between incompatible type systems in relational databases and object-oriented programming languages.   |
| Operating system          | refers to one of the standard web infrastructures on server side which provides an interface between the hardware and other software. This includes for instance Unix, Linux, FreeBSD, Microsoft Windows, and so on.   |
| Page load performance     | refers to one of the most important performance metrics for a mobile browser to specify the browser application's abilities in general, and thus to acknowledge the hardware and software appearances of the running mobile device. [Section 2.2.1]                  |
| Page loading speed        | refers to the average time period spent for a mobile browser on one single web page, or the sum of all time periods spent for a bunch of web pages. The page loading speed of a mobile browser is a measure to tell how fast the mobile browser's infrastructure,    |

such as parsers and engines, can parse, interpret and render web contents to the browser's user interface when responding to user requests for web addresses. [Section 2.2.1]

**Python** refers to a general-purpose high-level programming language. Its design philosophy emphasizes code readability. (Python Software Foundation, 1990-2010)

**Representational State Transfer (REST)**

refers to a style of software architecture for web applications and other distributed hypermedia systems. A web application framework is RESTful by providing convenience to the application development following REST style. (Fielding & Taylor, 2002)

**Rich internet application** refers to an online application working more like a desktop application than a traditional web application, e.g. webmail. RIAs are enabled by rich technologies such as Ajax which lead into faster response times and more interactive graphical user interface (GUI) elements, e.g. always visible floating menus, and controls, e.g. the dragging of GUI elements.

**Social networking** refers to online communities which are formed with social networking applications, which offer ways of finding people with similar interests, communicating with others, and expressing the user himself. Facebook is the most popular application worldwide.

**Usability** refers to a part of user experience that measures how easy and pleasurable a product or service is to use.

**User experience** refers to experience from a user which is positively achieved when service's features and design meet user's needs and expectations in a usable and pleasurable way.

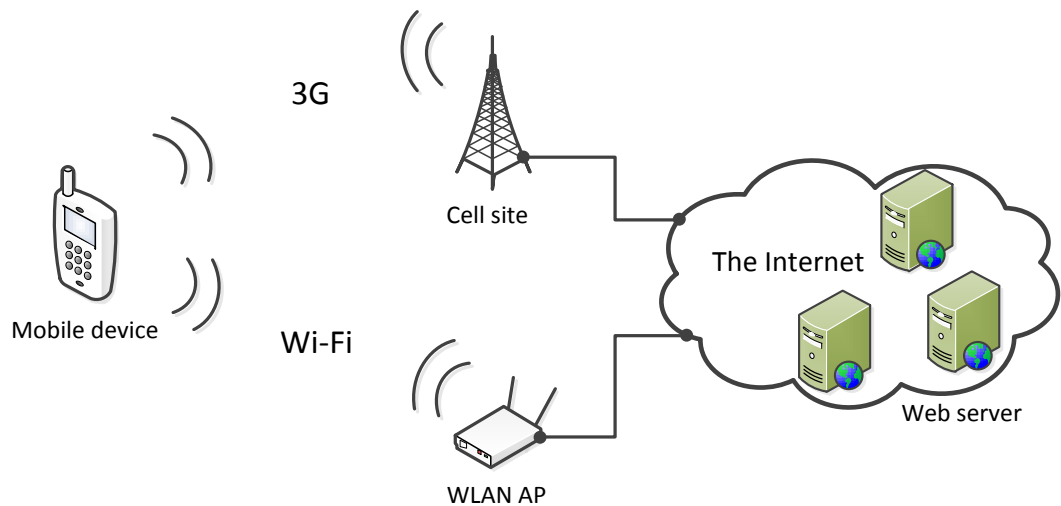
**Web 1.0** refers to a concept in order to present traditional web technologies being used before Web 2.0 concept appearing. The web technologies of Web 1.0 include for instance HTML, XHTML and CSS.

|                           |  |
|---------------------------|--|
| Web 2.0                   | refers to a concept for a collection of new technologies, applications, concepts, ideas, business strategies, and social trends in the web. Web 2.0 is more dynamic and more interactive than Web 1.0.                   |
| webapp                    | refers to a simple web application framework provided by Google App Engine by default. [Section 2.4.2]   |
| Web application framework | refers to a software framework that is designed to support the development of dynamic websites, web applications and web services. These include, for instance, Rails, Django, Spring, etc. [Section 2.3.3]              |
| Web server                | refers to one of the standard web infrastructures on server side which delivers contents using kinds of data transferring protocols over network. This includes for instance Apache, Microsoft IIS, Lighttpd, and so on. |
| Wiki                      | refers to a web-based tool for creating, modifying, and deleting web content collaboratively. Web-based encyclopedia Wikipedia is the best known online wiki application worldwide.                                      |

## 1. Introduction

With the explosive growth on personal usage of smartphones and handheld computing devices in recent years, especially after the launch of the revolutionary mobile device, *iPhone* (iPhone, 2011) introduced by *Apple* Incorporation (Apple, 2011) to public in early 2007, the proportion of mobile access to the Internet [Figure 1-1] keeps increasing dramatically compared to the traditional web access initiated from desktop computers and laptops. Consequently, people pay more and more attention to the software performance of *mobile browser* applications which are used most frequently (Gardner, 2010) on their handheld devices. However, instead of variously convenient software measurement tools to evaluate performance of *web browser* applications running on desktop computing environment, there are seldom such kinds of performance measurement tools available which are optimized specifically for those web browser applications integrated in mobile computing devices.

Different than running in capable hardware and software environments on desktop computers and laptops, the performance of web browser applications running on handheld devices is deeply affected by limited hardware capacities, restricted software execution environment, slow and unstable wireless connection, and shrunken interfaces for HCI (Human-Computer Interaction). Furthermore, it is difficult to completely execute a comprehensive performance evaluation for a mobile browser because there are a huge amount of performance metrics to be measured for comparison and benchmarking. Depending on various intentions of device manufacturers, vendors, network operators, technology media agencies and end users, a large number of human resources and computing resources are committed to measure and benchmark the performance of various brands and editions of mobile browsers together with hosted devices in the market in their own way. Increasing amounts of measurement facilities and environments are therefore established and served privately or in public all over the world although many of them are actually, as a matter of fact, redundant and unsustainable.



**Figure 1-1:** The infrastructure of mobile access to the Internet (Ye, 2010)

Explanation of the terms in the figure above:

- Mobile device** refers to smartphone or other handheld computing devices which is used to access the Internet via wireless connection.
- 3G** refers to the third generation of mobile telecommunication standards for wide-area wireless voice telephone, mobile Internet access, and other application services.
- Wi-Fi** refers to a narrow range of connectivity technologies including WLAN (Wireless Local Area Network) based on the IEEE 802.11 standards and etc.
- Cell site** refers to a base station consists of antennas and electronic communication equipment to create a cell in a cellular 3G network.
- WLAN AP** refers to an access point of WLAN connecting to a wired network.
- Web server** refers to both hardware and software help on delivering content to access through the Internet.

The *page load performance* of a mobile browser is defined as the time period spent by a browser application to completely present the contents on a web page to the browser's display area immediately after acknowledging a new request submitted by user. The average time period spent for a mobile browser on one single web page, or the sum of all time periods spent for a bunch of web pages, is called *page loading speed*. The page loading speed of a mobile browser is a measure to tell how fast the mobile browser's infrastructure, such as parsers and engines, can parse, interpret and render web contents to the browser's user interface when responding to user requests for web addresses. Compared to some other performance metrics of a mobile browser, such as initial *start-up speed* or *JavaScript* execution speed, page loading speed is more convenient and more frequent to be observed by users, who used to access the Internet and switch back and forth within a bunch of hyperlinks to web sites. Consequently, the page load performance of a mobile browser is more intuitive and relevant to the *user experience* of the browser.

However, like with the evaluation of other performance metrics of mobile browsers, there are no conventions for the measurement of page loading speed which are available in public documentation. For example, the usage of stopwatches to manually measure the page loading time of a web page is simple but lacks reliability and scalability for large-scale and continuous performance evaluation. Meanwhile some development and testing teams of mobile browsers working for mobile device manufacturers, such as Nokia's Maemo Browser teams, utilize high level SQA (Software Quality Assurance) process and more standardized practices to evaluate the page load performance than individuals, which makes the measurement process and data storage automatic. The teams had to commit a lot of hardware and human resources at the beginning to establish a specific measurement environment. However, the established environment consisting of valuable hardware and software commitments would not be portable to fit the performance evaluation demand on any other mobile device than the one in investment.

### **1.1. Purpose of the Study**

In order to help the current situation of performance evaluation of mobile browsers, this study focuses on finding out an efficient and accurate measurement process to determine the page loading speed of mobile browsers while avoiding unnecessary manual intervention and dependencies to the measurement environment. The purpose of this study is to identify a relatively accurate and efficient process to evaluate the execution performance of a mobile browser, and to implement a measurement application



accordingly in the experimental part. The study concentrates on one of the critical performance metrics, the page loading speed, instead of trying to cover every possible performance metric of a mobile browser, because page load performance of a mobile browser is more intuitive and significant than other metrics relevant to the browser user's experience.

Furthermore, as a part of the implementation of the measurement application in the experimental part, this study tries to eliminate the complexity of setting up the measurement environment by introducing cloud computing services as the hosting platform and transferring the execution environment to the measurement application which is thus publicly accessible anytime anywhere from the Internet as most modern web 2.0 applications are.

In order to measure the page loading speed of mobile browsers, a set of web pages should be defined beforehand as the standard responding data for browser's page loading requests. In the experimental part of the study, the category of web pages which are used in the measurement process as the basis for detecting page loading speed of mobile browsers is collected from a well-known Internet traffic and page ranking service provider, Alexa.com, which provides a reliable list of global top sites based on their traffics.

There is no standard data of absolute values available to judge a browser's page load performance. Instead, continuous measurement of page loading time and data collecting is used to provide reliable reference to determine possible performance advancement or regression after changes have been made to the application implementation or execution dependencies of the mobile browser. This recorded data on page loading time is also useful for weighing up the performance gap between various mobile browsers from different manufacturers.

The engineering process for design and implementation of such an online measurement model for mobile browsers' page loading speed is similar as to create web 2.0 applications which are more and more popular on the Internet. The UI (User Interface) design specific for mobile devices and some Web 2.0 techniques, such as Ajax, are also beneficial for the implementation to improve the online experience of mobile users.

## 1.2. Research Questions

The research questions of this study mainly focus on finding out a measurement process for the page loading speed and according practices to ease the evaluation on the page load performance of mobile browsers in an accurate and efficient way. The research also focuses on the benefits and defects brought by using the cloud computing services as the implementation environment and hosting platform of the experimental work. Through the observation on practical usage of the implemented online application in the experimental part of the study, this research tries also to find out possible factors which affect the page load performance of a mobile browser. The research questions are presented here in an order of importance.

### **1st Question: How to evaluate page load performance of mobile browsers accurately and efficiently?**

This study concentrates on identifying an optimized measurement model and according operational process to determine the page loading speed of a mobile browser, which should not only be easy to operate by measurement tool users, and proper on collected results, but also widely applicable to most kinds of mobile browsers on various mobile devices available in the market no matter on their manufactures or vendors.

The structure of a mobile browser will be introduced and discussed from part to part in order to understand the procedure for a mobile browser to execute a page loading operation to a web page and find out possible measurement approaches. Both advantages and disadvantages of these approaches will be presented and discussed during the study.

By selecting the measurement model as a fundamental part of the results of our research, [Section 4.1.1] which is based on web 2.0 concepts and techniques, an according operational process of the measurement is specified and described in detail. The main objectives are to indicate:

- What should be considered for a common measurement process that would determine the page loading speed of a mobile browser?
- How to identify a practical measurement model from alternatives?

Looking through the factors which influence measurement accuracy and efficiency in page loading speed of mobile browsers helps in improving other performance evaluation processes which are in use in the mobile industry.

## **2nd Question: What are the benefits and drawbacks of implementing the measurement model on a cloud computing platform?**

In the experimental part of this study, a software application is designed, implemented, tested and deployed to examine and verify the usability of the measurement model which is determined as the answer of previous question. Different from traditional software engineering practices, the experimental work is implemented and deployed on a cloud computing platform. There are many cloud computing services available on the Internet served by various providers, such as EC2 (Elastic Compute Cloud) from Amazon, and App Engine from Google. The selection of the cloud computing platform in the experimental work is Google's App Engine, as it is the only cloud service which is free of charge to get started without limitation on time of usage, and its data traffic restriction is sufficient to sustain all of the data flows generated from the measurement practices in the experimental work.

Compared to software applications running in local environment, online applications based on cloud computing services are similar as modern web 2.0 applications popular on the Internet nowadays, which is accessible online all the time and ready to be used from anywhere in the world. As the experimental part of the study achieved the design and implementation of the measurement application within cloud computing services, this study tries to determine the influences brought by this kind of practices. Therefore, this thesis focuses also on the following topics:

- What are the benefits and drawbacks of introducing a cloud computing platform into the experimental work?
- What are the drawbacks of deploying the measurement tool onto a cloud computing platform?

Determining the benefits and drawbacks of implementing and deploying the performance management tool on a cloud computing platform may help to find out more possibilities on the design and development of a more practical measurement environment for device manufacturers and end users of mobile browsers.

## **3rd Question: How to improve the page load performance of a mobile browser?**

After discussing the factors that influence the measurement process and the practical implementation of a measurement application, there will be some related observation on the possible factors which influence the page loading speed of a mobile browser. Some

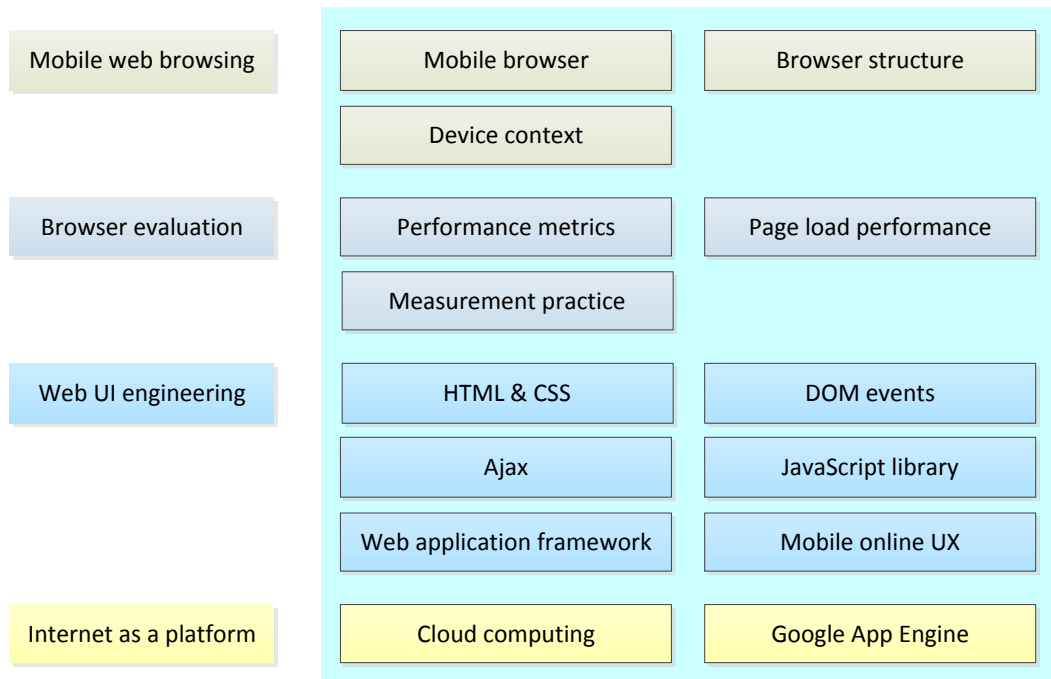
come from the content of the web pages visited on page loading operation while others come from the specific features or components of the mobile browser, and yet more from the adopted measurement process and practical environment and software tool chains. The factors observed from the operation practices by using the implemented online management application in the experimental part of the study would be part of the answer of this question. The purpose is to find out:

- How does the design of a web page influence its page loading time in a mobile browser?
- What features or components of a mobile browser may influence its page load performance?

Determining the factors that affect the page load performance of a mobile browser is beneficial not only for the web page designers and developers to create faster web sites for mobile access, but also for end users to achieve a better user experience when using mobile devices to surf the Internet in daily occasions.

### **1.3. Structure of the Thesis**

The structure of this Master's thesis is the following: the literature overview concentrates on determining the concepts of mobile browser [Section 2.1], page load performance [Section 2.2] and cloud computing [Section 2.4], as well as the characteristics and features provided by the practical development environment and hosting platform, Google App Engine [Section 2.4.2]. The research chapter describes the research methods using in the experimental part of the study, and lists issues found, as well as considerations and possible solutions which emerged during the research. [Section 3] The results of the research [Section 4.1] and practical measurement results for verification on the experimental part [Section 4.2] are located in the results chapter. In the discussion chapter, the predetermined research questions are answered [Section 5.1] with benchmarks to several popular performance measurement applications [Section 5.2]. In the last chapter of the thesis, the conclusions of the research, defects and possible directions of future research are discussed. [Section 6] The developed source code of the implementation in the experimental part of the study is presented in the appendices.



**Figure 1-2:** Scope of the study

Explanation of the terms in the figure above, from top to bottom and left to right:

- Mobile browser** refers to a web browsing software application designed for use on various mobile devices, such as smartphones or tablet computers. [Section 2.1.1]
- Browser structure** refers to the software architectural structure of a modern mobile browser clarified in high level. [Section 2.1.2]
- Device context** refers to systematic contexts to interact with the executing browser on a mobile device, which may include hardware capacities, software system, network connections and etc.
- Performance metrics** refer to possible performance indicators being able to be measured and compared for performance evaluation across various mobile browsers. [Section 2.1.3]
- Page load performance** refers to the performance of a mobile browser to respond to page loading requests, which is commonly indicated by the page loading speed of specified web pages as references. [Section 2.2.1]

|                                  |   |
|----------------------------------|---|
| <b>Measurement practice</b>      | refers to the measuring process and practical operational steps when doing the performance measurement of a mobile browser. [Section 2.2.2]   |
| <b>HTML &amp; CSS</b>            | refer to acronyms for HyperText Markup Language and Cascading Style Sheets, which are two basic construction components of web pages.   |
| <b>DOM events</b>                | refer to an acronym for Document Object Model events, which allow JavaScript language to listen and handle inside a HTML document being manipulated by a web browser.   |
| <b>Ajax</b>                      | refers to an acronym for Asynchronous JavaScript and XML, which is a group of inter-related web development techniques used to create interactive web applications on the client side. [Section 2.3.1]  |
| <b>JavaScript library</b>        | refers to a set of pre-written JavaScript controls which provides convenience to the development of JavaScript-based applications. [Section 2.3.2]  |
| <b>Web application framework</b> | refers to a software framework that is designed to support the development of dynamic websites, web applications and web services. [Section 2.3.3]  |
| <b>Mobile online UX</b>          | refers to user experience of an online application for users of mobile browsers.  |
| <b>Cloud computing</b>           | refers to a computing capability that provides an abstraction between the computing resource and its underlying technical architecture, enabling convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction. [Section 2.4.1] |
| <b>Google App Engine</b>         | refers to a set of Internet services consisting of web application development environment and hosting platform provided by Google. [Section 2.4.2]   |

## 2. Literature Overview

The page load performance of a mobile browser is related to performance and modularity of both software and hardware components. Additionally, bandwidth of the wireless network connection and performance measuring approaches may also affect obtained measurement results. In this chapter some general definitions of mobile browser are discussed. Several commonly used measurement approaches for performance evaluation on mobile browsers are then described in brief. In the last part the concept of cloud computing services and the selected instance in the experimental part of the study, Google App Engine, are also introduced.

### 2.1. Mobile Browser

In this section the concept of mobile browser is defined and explained in detail. In order to identify the possible factors in the components of the software structure of a mobile browser which could affect the page load performance of the application, the general structure of a mobile browser is introduced in high level. And in the final part several performance metrics popularly used in public measurement practice are introduced and discussed.

#### 2.1.1. Definition of Mobile Browser

A mobile browser, also called a micro-browser, mini-browser or wireless Internet browser, is a web browsing software application designed for use on a mobile device, such as a mobile phone, PDA, tablet computer or one of any other kinds of wireless handheld devices. (Ari Jaaksi, 2002) Mobile browsers are commonly optimized depending on the hardware capabilities of hosting handheld devices, which considerably means weak computing power, limited memory capacity, insufficient electric power provided by batteries, low and unstable bandwidth of wireless connection, restricted HCI (Human-Computer Interaction) interface, and compact display area, to present web content quickly and properly in best efforts.

In order to achieve the similar user experience as traditional web browser applications running on desktop computing environment, a number of mobile browsers developed in early ages had to restrict some useful functionalities and storage consumption, thus stripped away some non-trivial features, such as advanced CSS (Cascading Style Sheets) support, JavaScript interpretation, and allowance of multimedia plugins which are playing critical roles in modern social network services supported by Web 2.0 technologies.

Gradually, these features are implemented in mobile devices since dramatic evolutions on their hardware capabilities keep going all the way.

Nowadays, mobile browsers play more and more important roles in the field of telecommunication and information technology industry. For mobile operators, mobile browsers introduce an extensive and fully customizable data services platform and consistent end user experience across multiple device types; For device manufacturers, mobile browsers provide a modularized architecture, easy integration, extensibility, compact footprint, and a reliable and proven solution on all major platforms and operating systems; For end users, they offer the fastest, richest and most intuitive, full-Internet browsing experience and services on mobile devices; And for contents providers, they contribute a rich content rendering platform for embedded devices.

### **2.1.2. Structure of Mobile Browser**

In order to identify a mobile browser's performance, the software realization of a mobile browser needs to be clearly understood. Derived from fully functional web browser applications running on desktop computing environment, most modern mobile browsers are constructed following software architectures similar to those of their predecessors. As the main components in the high level structure of these mobile browsers are kind of similar, they will be introduced here in a unified way. A general structuration of mobile browsers is shown in the graph. [Figure 2-1] It comprised eight major subsystems, which consist of User Interface, Browser Engine, Rendering Engine, Networking subsystem, JavaScript Interpreter, XML Parser, UI Backend, and Data Persistence subsystem, together with the dependencies between them. (Alan Grosskurth, 2006)

#### **User Interface**

The User Interface (UI) subsystem is the layer between the user and the Browser Engine. It provides features such as toolbars, visual page-load progress, smart download handling, preferences, and printing. It may be integrated with the desktop environment to provide browser session management or communication with other desktop applications.

#### **Browser Engine**

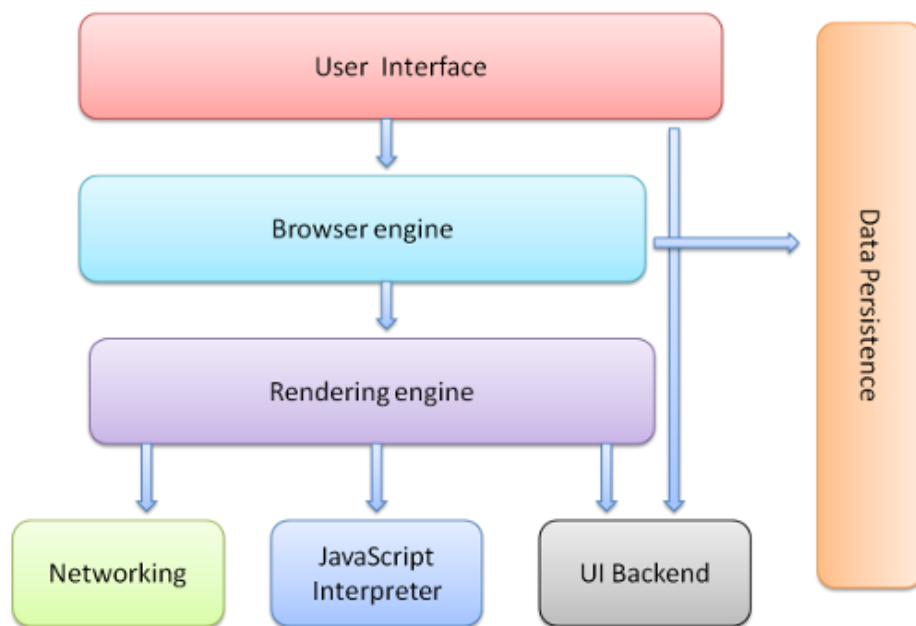
The Browser Engine subsystem is an embeddable component that provides a high-level interface to the Rendering Engine. It loads a given URI (Uniform Resource Identifier) and supports primitive browsing actions such as to go forward, back, and to reload. It provides hooks for viewing various aspects of the browsing session such as current page load process



and JavaScript alerts. It also allows the querying and manipulation of Rendering Engine settings.

### Rendering Engine

The Rendering Engine subsystem produces a visual representation for a given URI. It is capable of displaying HTML and XML (Extensible Markup Language) documents, optionally styles with CSS, as well as embedded content such as images. It calculates the exact page layout and may use “reflow” algorithms to incrementally adjust the position of elements on the page. This subsystem also includes the HTML parser, which is often tightly integrated with the Rendering Engine for performance reasons and can provide varying levels of support for broken or nonstandard HTML. This tight integration is the result of a design decision and seems to be a common feature of web browser architectures.



**Figure 2-1:** High level structure of web browsers

### Networking

The Networking subsystem implements file transfer protocols such as HTTP (HyperText Transfer Protocol) and FTP (File Transfer Protocol). It translates between different character sets, and resolves Internet media types for files. It may implement a data transfer cache of recently retrieved resources.

### **JavaScript Interpreter**

The JavaScript Interpreter evaluates JavaScript code, which may be embedded in web pages. JavaScript is an object-oriented scripting language developed originally by Netscape (Netscape Communications, 2008). Certain JavaScript functionality, such as the opening of pop-up windows, may be disabled by the Browser Engine or Rendering Engine for security purposes.

### **XML Parser**

The XML Parser subsystem parses XML documents into a DOM (Document Object Model) tree. This is one of the most reusable subsystems in the architecture. In fact, almost all browser implementations leverage an existing XML Parser rather than creating their own from scratch. Although arguably less important to the functionality of the system, the XML Parser is a generic, reusable component with a standard, well-defined interface.

### **Display Backend**

The Display Backend subsystem provides drawing and windowing primitives, a set of user interface widgets, and a set of fonts. It may be tied closely with the operating system.

### **Data Persistence**

The Data Persistence subsystem stores various data associated with the browsing session on disk. This may be high level data such as bookmarks or toolbar settings, or it may be low level data such as cookies, security certificates, or cache.

### **2.1.3. Performance Metrics of Mobile Browser**

There are a great amount of performance metrics to evaluate a mobile browser among other alternatives in current market. Depending on different intentions from device manufacturers, vendors, network operators, technology media agencies and end users, various performance evaluation metrics are identified and adopted to measure and benchmark the performance of various brands and editions of mobile browsers together with the located mobile devices in the market in people's own way. As the same as a web browser running on powerful computing environment, these performance metrics of a mobile browser can be separated into several aspects based on user requirements.

### **Startup speed**

Browser's Startup speed means the time a browser application uses to launch when the software running environment is boot up and ready to serve, which starts commonly from a

single or double click on its application icon to the visible view area of the pre-defined first page being correctly presented to the display on the mobile device for user.

### **JavaScript speed**

JavaScript, also known as ECMAScript, is a dynamic, weakly-typed, object oriented scripting language that is primarily used in the form of client-side script interpreter and executor as part of a web browser in order to provide enhanced user interfaces on dynamic websites. Combined JavaScript with asynchronous data transferring interface provided by most web browsers and redefined data packaging convention for streaming, Ajax techniques provide the opportunities to create much better user experiences than ever, on social network websites and other web applications.

The speed of JavaScript interpreter deeply influences the speed for web browser to render websites, especially to Web 2.0 websites which are implemented with comprehensive scripts to present interactive effects on user interface and to manipulate the data across the browser and remote servers. Faster JavaScript execution time means that those web sites which use Ajax heavily, such as Digg.com, and most of online software applications, such as Gmail from Google, will be more responsive to user actions.

### **DOM selection speed**

The DOM (Document Object Model) is a platform-neutral and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents. The document can be further processed and the results of that processing can be incorporated back into the presented page. The faster a browser application can select elements in a web page, the more responsive it is on asynchronous page updates, which most Web 2.0 online applications heavily rely on.

### **Page loading speed**

The page loading speed, which is also called as page loading time, is the total time a mobile browser takes to load one single web page, or the sum of all time spent for a bunch of web pages. The page loading speed is a measure to tell how fast a browser application's infrastructural components, such as parsers and engines, can parse, interpret and render web contents to User Interface of the browser when responding to requests for web addresses from user.

### **Average CPU and memory usage**

The usage of CPU (Computer Processing Unit) power and physical memory reveals how much system resources a browser application needs to be performing. Higher CPU utilization or more memory occupation indicates more resource consumption by a browser application, which is a negative factor when evaluating the performance of a mobile browser.

### **CSS rendering speed**

The CSS (Cascading Style Sheets) is a style sheet language used to describe the presentation semantics, such as the look and formatting, of a document written in a markup language, which is HTML (HyperText Markup Language) or XHTML (Extensible HyperText Markup Language) in most common applications. A mobile browser with faster CSS rendering speed would have faster page response time when loading a web page. This speed is commonly measured by detecting the page loading duration which a browser application spent on completely rendering a specific-designed web page consisting of a large number of table-to-div conversion requested from CSS formatting statement.

### **Browser cache performance**

The browser cache is a mechanism for a web browser application to employ the temporary storage of web documents, such as web pages and images, which have ever been accessed to reduce connection bandwidth usage and perceived lag on page loading. The browser cache is also beneficial to visited web sites as it decreases the workload for the servers of those web sites to respond to duplicated user requests.

## **2.2. Page Load Performance**

A user of a mobile browser always hopes both the browser application itself and the web pages he tries to surf to respond to his operation instantly. However the fact is quite complicated in real world. In order to make the web work faster and faster on mobile devices, many online service providers are doing their best on increasing the accessibility of the entrance page to their network services. Furthermore, mobile device manufacturers are putting their efforts into creating more powerful hardware with higher performance browser applications integrated in the software stacks on devices. The page load performance of mobile browsers is affected by all of these dynamic factors.

### **2.2.1. Definition of page load performance**

The page load performance is one of the most important performance metrics for a mobile browser to specify the browser application's abilities in general, and thus to acknowledge the hardware and software appearances of the running mobile device. It is commonly indicated by the time period spent by a browser application to completely present visual contents on a web page to the browser's display area immediately after acknowledging the initial request submitted by user.

The time period spent for one single web page, or the sum of all time periods spent for a bunch of web pages, is called page loading time, as known as page loading speed. The page loading speed of a mobile browser is a measure to tell how fast the mobile browser's infrastructure, such as parsers and engines, can parse, interpret and render web contents to the browser's user interface when responding to user requests for web addresses. Compared to some other performance metrics of a mobile browser, such as initial start-up speed or JavaScript execution speed, page loading speed is more convenient and more frequent to be observed by browser users who used to access the Internet and switch back and forth within a bunch of hyperlinks to web sites, and thus is more intuitive and relevant to the user experience.

### **2.2.2. Measurement of page load performance**

There is no absolutely standardized data of page loading speed available to judge a browser's page load performance. Due to latency differences that occur with variable site traffic and server load, the measurement results of the page loading speed cannot be treated as the intuitive basis for evaluation. Instead, they should be interpreted with caution. However, continuous measurement and collections of recorded values are useful to provide reliable reference to determine possible advancement or regression of performance after changes in source code or runtime environment of the browser application. The recorded measurement results are also used to weigh up the performance gap among alternative mobile browsers which come from different manufacturers.

When a user submits a web address as a request to a browser application, which commonly means to input a URL (Uniform Resource Locator) in browser's Location Bar and to press the "Go to" button next to the Location Bar, the browser needs to do a series of actions to fetch the specified content of the web page remotely through network connection, and then parse and render it properly onto the user interface. During this comprehensive process, not only the performance of the mobile browsing application but also the hardware

capacities of the mobile device and the connection throughput between the browser and the server play important roles on the dependencies of the page loading speed.

## **2.3. Utilities for Web Application Development**

During the implementation of the experimental part of the study, some modern web programming techniques and helpful tools are applied to create a practical online measurement application to specify and verify the generated measurement process of mobile browser's page loading speed in research. For instance, Ajax is used in the application to transfer generated data back and forth between the measuring browser and the application server located on Google's cloud computing platform. The jQuery library, which consists of a great number of useful JavaScript controls to provide convenience to JavaScript related programming, is also used to ease the workload of coding and to enhance user experience to the online experimental application. Additionally, the default web application framework integrated in Google's App Engine services, webapp, established the standardized development environment and the deployment structure for the actual implementation. These concepts are therefore explained and discussed in the following sections.

### **2.3.1. Ajax**

Ajax (Asynchronous JavaScript and XML) is a group of inter-related web development techniques used to create interactive web applications on the client side. With Ajax, web applications can retrieve data from the server asynchronously in the background without interfering with the display and behavior of the existing page. (Garrett, 2005)

Ajax is not a solo technology but a group of technologies instead. Ajax uses a combination of HTML (HyperText Markup Language) and CSS (Cascading Style Sheets) to mark up and style information. The DOM (Document Object Model) is accessed with JavaScript to dynamically display and to allow the user to interact with the information presented. The use of Ajax techniques has led to an increase in interactive or dynamic interfaces on web pages. Despite the name of Ajax, the use of XML (Extensible Markup Language) is not actually required for data interchange. Instead, JSON (JavaScript Object Notation) (JSON, 1999) is often used as an alternative format for data interchange, although other formats or plain text can also be used.

In many cases, related pages on a website consist of much content that is common between them. Using traditional methods that content would have to be reloaded on every request. By using Ajax, however, a web application can request only the content that needs

to be updated, thus drastically reducing bandwidth usage and page load time. Additionally, the use of asynchronous requests allows the client's web browser UI (User Interface) to be more interactive and to respond quickly to user inputs, and sections of pages can also be reloaded individually. Users may perceive the application to be faster or more responsive even if the application has not changed on the server side. Furthermore, using Ajax reduces connections from the client to the server, since scripts and style sheets only have to be requested once.

Despite all of these benefits, in practical situations, Ajax interfaces are often harder to develop due to their dynamic nature when compared to static pages. To reduce the barriers to Ajax development for web developers, most modern *JavaScript Libraries* provide an *Ajax framework* and corresponding utilities.

### 2.3.2. JavaScript library

The JavaScript library is a set of pre-written JavaScript controls which provides convenience to the development of JavaScript-based applications, especially for Ajax and other web-centric technologies. It was developed to meet the expended demands for JavaScript with the rise of Web 2.0 concepts and their implementations.

By combining most of common operations invoked through JavaScript functions and common Ajax invoking functions, the JavaScript library allows developers to concentrate more upon the web user interface behaviors, without being disturbed by the compatibility issues caused by the client-side browser difference. For instance, the jQuery JavaScript library (jQuery, 2006) supports browsers which consist of Mozilla's Firefox version 2.0 and above, Microsoft's Internet Explorer version 6 and above, Apple's Safari version 3 and above, Opera's Opera version 9 and above, and Google's Chrome version 1.0 and above. (jQuery, 2010)

The JavaScript library is even chosen to be embedded into some web application frameworks as a component of those to provide further convenience to web developers. For instance, the web application framework Rails (Rails, 2004-2010) natively supports a JavaScript library which is called Prototype (Prototype, 2006). There are also some web application frameworks which do not embed a JavaScript library, but are easy to work with one or several JavaScript libraries together depending on their loose design and modular structure. (Bennett, 2006)

### 2.3.3. Web application framework

A web application framework is not only a web-based software development framework but also a reusable, skeletal, semi-complete modular platform. It is designed to support the development of dynamic websites, web applications and web services. Additionally, it is commonly allowed to be specialized to serve as a custom web server via HTTP(S) protocol. (Shan & Hua, 2006) The framework aims to relieve the overhead associated with common activities performed in web development. There are numerous web application frameworks available which associate different programming languages for each. These include, for instance, Rails framework which is built on top of Ruby language, (Rails, 2004-2010) Django built on Python language, (Django Software Foundation, 2005-2010) and Spring on Java. (SpringSource, 2004-2010)

Most web application frameworks were constructed with built-in modules to provide a range of services by following industrial standard software architectural patterns and coding guide principles which had been defined and broadly used in software engineering. For instance, the famous software architectural pattern MVC (Model-View-Controller) is always integrated to gain the benefits from the isolation of business logic from the user interface and easier code maintenance; (Selfa, Carrillo, & Del Rocio Boone, 2006) A software coding principle, DRY (Don't Repeat Yourself) is widely respected in the framework implementation; (Thomas, 2003) A software design philosophy, convention over configuration (Miller, 2009) and a software architectural style specific for web application design, REST (Representational State Transfer) are also carefully included in the implementation of most web application frameworks as well. (Fielding & Taylor, 2002) A mature and productive web application framework will be in-complete when lacking of any of these characteristics.

In client-server structure, a web application framework plays the role of the middleware receiving all requests from the client, querying data from database, and responding back to the client with the generated results, which are commonly rendered web pages. Built with the design patterns, development principles and architectural styles by nature, most web application frameworks provide various features to simplify the work for web developers. These features aim to become helpers on a number of matters from rendering web pages, user authorization, fetching data from a database, to reducing the time consumed by a single query. (DocForge, 2010)



## 2.4. Cloud Computing

The concepts of cloud computing and related services are nowadays getting more and more familiar to online service providers and Internet users. Cloud computing services provide secure access to all online content, such as web applications and data, from any network device. [Figure 2-2] Running online software applications on a platform provided by cloud computing services brings a lot of benefits against traditional software applications running in local computing environment. In the experimental part of this study, the practical implementation of the measurement tool is programmed and hosted within a cloud computing platform, App Engine provided by Google. Therefore the concepts of cloud computing and Google App Engine are explained and discussed here.



**Figure 2-2:** Cloud computing visual diagram

### 2.4.1. Definition of cloud computing

Cloud computing is a computing capability that provides an abstraction between the computing resource and its underlying technical architecture, enabling convenient, on-demand network access to a shared pool of configurable computing resources that can be rapidly provisioned and released with minimal management effort or service provider interaction. (Mell & Grance, 2009) It is often considered as web application development and technical manipulation based on the Internet.

Instead of a traditional client-server structure, cloud computing abstracts the details of control over technology infrastructure from users and sets up a new supplement, consumption, and delivery model based on the Internet. Furthermore, it involves also dynamical scalability and resource virtualization as an Internet service.

#### **2.4.2. Google App Engine**

Nowadays, there are quite some cloud computing services available in the business market. One of them is Google App Engine. Google App Engine is a platform for developing and hosting web applications in Google's infrastructure. With the power of Google-managed servers and data centers, web applications implemented and deployed on App Engine are migrated with cloud computing technology seamlessly, which assures the stability and flexibility of its hosting service globally. (Google, 2008)

Google App Engine was designed and implemented to be programming language independent to attract web developers from different technical backgrounds. When it was first released as a beta version in April 2008, Google App Engine supports Python language only. And one year later the support of Java language was added. After the Java support was added in early 2009, theoretically, any programming language which is supported by the JVM (Java Virtual Machine) is available to be used for coding with App Engine nowadays.

Google App Engine was designed to work with most web application frameworks which are written in any one of its supported programming languages. For Python programmers, Google App Engine provides also a simple web application framework named as webapp, (Google, 2008) which inherited some advantages and characteristics from another powerful Python-based web application framework, the Django web framework (Django Software Foundation, 2005-2010) . With this built-in web application framework and other unique features, Google App Engine provides a complete, feature-rich environment for implementation and online publication of web applications.

### 3. Research

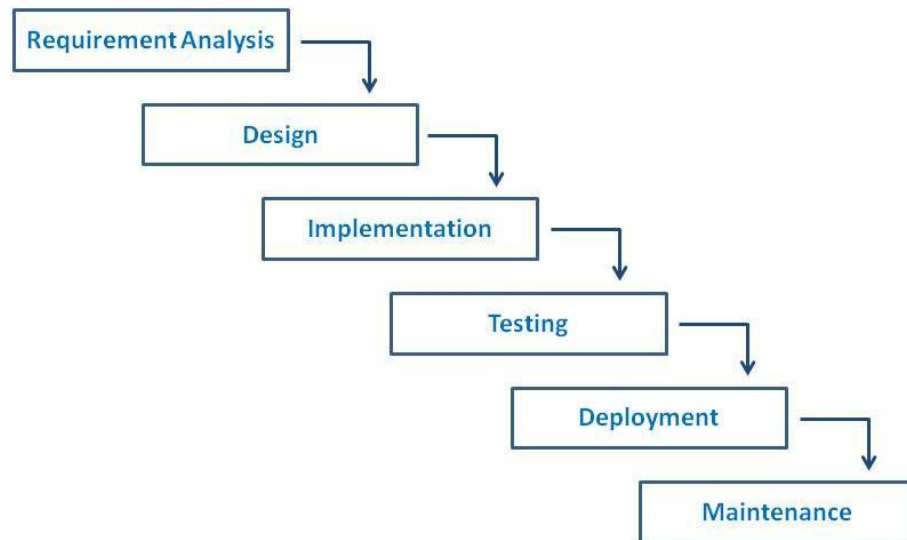
In order to find out an accurate and efficient method to measure page loading speed of mobile browsers, a measurement model and according measurement operational process should be designed, implemented and verified based on practices of evaluation activities. In this chapter, the research method which has been applied to design and programming on the implementation in the experimental part of the study is described. Each step of the research process has been clarified and discussed separately with details of consideration, possible alternative solution, and final decision on implementation. Some coding practices of the research issues are also presented in this chapter.

#### 3.1. Research Method

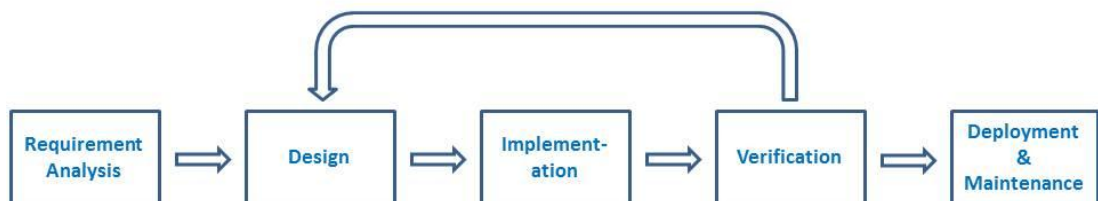
Because the aim of the experimental part of the study is to create an online software application, as similar as to create a local software application, to measure and record the page loading speed of mobile browsers, the research method in the study tries to follow the standard development process popularly used on industrial software engineering. As the workout of the code implementation needs to be verified with actual measurement activities and modified based on the verification when necessary, some steps of the software engineering process, such as Design and Implementation, should be allowed to be revisited. So the traditional Waterfall model [Figure 3-1] does not fit in this work. Instead, a model of agile software engineering process [Figure 3-2] is applied all the way in the research and the programming work in the experimental part of the study.

The agile software engineering process applied in the experimental part of the study supposed to have six steps, which consists of requirement analysis, design, implementation, verification, deployment and maintenance. As the implementation of the experimental part will be developed and hosted on a cloud computing platform, the steps of deployment and maintenance, which get benefits on hosting the completed application on the cloud computing services, are therefore simplified a lot so that unnecessary to be discussed separately in this study. So there is only one section for the discussion on these two steps.

The outline of the description of each step of the agile software engineering process applied in the experimental part of the study is listed below.



**Figure 3-1:** The Waterfall model of software engineering process



**Figure 3-2:** The agile model of software engineering process applied in this study

### Requirement Analysis

The Requirement Analysis phrase is the first step of the software engineering process in order to identify the requirements of the purpose. During the research work of this study, the analysis will collect and classify the initial needs and conditions for creation on the measurement process of the page loading speed of mobile browsers. The concern about project budgets and the portability of the measurement process will be mentioned and discussed. Additionally, a comparison between the manual approach and the automatic measurement method will be presented and the choice will be made.

## **Design**

In order to illustrate practical measurement models, the fundamental modular components of the measurement process will be identified at the beginning of the design phase. And then several possible measurement models will be illustrated for discussion on both advantages and disadvantages. Each part of the discussing measurement models will be justified with possible alternatives. In the last part of the design the elements of the user interface of the experimental online measurement application would be drafted and optimized.

## **Implementation**

The implementation will try to make the selection on standard development tools at the beginning. And then the specified measurement model will be implemented as the foundation work of the experimental part of this study. Some concerns and tricks useful in the implementation phase will also be presented and explained.

## **Verification**

In the verification phase a category of web pages will be defined and collected as the reference to support the measurement results of the page loading speed. Several different models of mobile devices will be used as the testing objects for data collection on measurement results.

## **Deployment and Maintenance**

This last engineering phase of the research process will deploy the implementation onto the selected cloud computing platform, with the help from convenient SDK (Software Development Kit). The benefits of hosting the experimental work on a cloud computing platform will also be specified.

## **3.2. Requirement Analysis**

Instead of trying to do the evaluation on all possible performance metrics of mobile browsers which are integrated in mobile devices available in the market, this research focuses only on measuring and evaluating the page load performance of mobile browsers because of its importance to user experience. [Section 2.2.1] The main purpose of this research is to identify a practical operational process to execute the measurement on the page loading speed of web browsing applications running on mobile devices. In order to prove the measurement process is easy to execute and adaptable for possible requirement changes, a practical software application should be designed and implemented as the

experimental part of the study. Based on this purpose, the main concerns of the requirement are listed below.

### **3.2.1. Functionality**

The implemented software application in the experimental part of the study should be able to detect, to record and to review the data of measurement results generated after each measurement rounds being executed in the measurement process. The detection approach should be automatically executed without manual intervention in order to achieve better accuracy on generated data. All generated data needs to be stored in database consistently and available to follow-up manipulation at any time in the future. The establishment and configuration of relevant measurement environment should be easy to operate and cost saving.

Since the measurement process concentrates on web browsing applications running on mobile devices, the implemented measurement application should provide a friendly UI (User Interface) specific for mobile access. It means that the color contrast of UI should be clear enough and the font size should be easy to read through compact display area on mobile devices.

### **3.2.2. Portability**

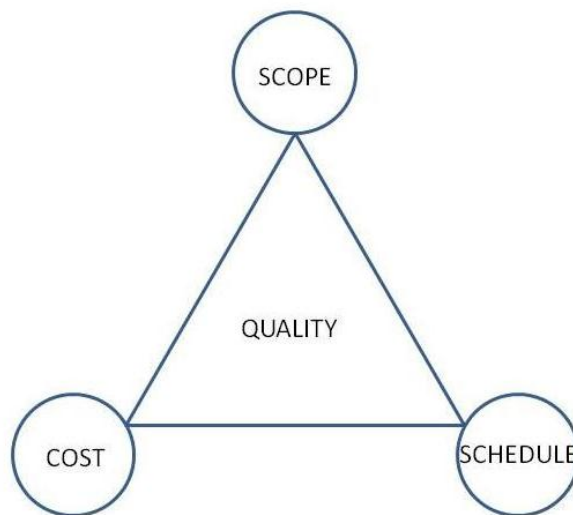
The solution of measurement process should be as much as possible to be portable to various kinds of mobile devices produced by different mobile device manufactures, no matter of differences on hardware capacities or on software operation systems.

The implemented software application should provide a convenient approach to measure the page loading speed of various mobile browsers, which may come from different mobile device manufactures, to ensure the measurement process is portable and reusable onto most kinds of mobile devices.

If the implemented measurement tool is a traditional software application running in local environment, the implementation and related hardware or software configuration should be easy to replicate in order to be deployed to other environment if needed. However if the implemented work is an online software application which is serving in public domain area as a normal Internet application, the requirement for replication would be significantly reduced accordingly.

### 3.2.3. Cost restriction

From the point of view on general project management theory for software engineering process, cost is one of the most fundamental factors which affect the quality of the output at the end of a project. [Figure 3-3] However, in the experimental part of this study the budget in fact does not exist at all. Within this none budget as a premise requirement of the research, the research phrases of the engineering process for the experimental work in this study, such as development, verification and deployment phrases, should avoid any possible expenses. This would inevitably affect the decision on the selection of optional software development environment, programming tools, and hosting platform of the experimental work. It would also influence the functional complexity of the implementation in the experimental work and particularly relevant programing practices.



**Figure 3-3:** The Projects Management Triangle

### 3.2.4. Manual vs. automatic

There are some individual users of mobile devices keeping trying to measure the page loading speed of a mobile browser manually. This kind of manual operation of measurement process is easy to execute without any preparation or environment setup except a stopwatch in hand. For instance, starting up the mobile browser on a running mobile device with available network connection, typing in a new web address as the reference to the location bar of the browser UI, and then clicking the “Go” button next to

the location bar to launch the page loading operation of browser application, while pressing the stopwatch in the other hand to start timing at exactly the same time. The stopwatch is then pressed again when the user observed at the first moment that the requested web page is completely rendered in the visible display area of the browser application.

Although the operation process is intuitive and easy to execute, it is obvious to tell that the measurement results of such a kind of manual practice lack of either reliability or accuracy. There are too many unreliable factors being able to infect the measurement results during the execution. Such a kind of measurement practice might be useful to do a casual comparison between two or three different mobile devices, but is unsustainable to be adopted in productive environment.

In the productive environment there exists another practice to do the measurement on the page loading speed of a web browser application, which captures the finishing moment of a page loading behavior automatically by detecting a specific DOM (Document Object Model) event, “onload”, which would be dispatched out by JavaScript when the content of a web page is completely loaded in the browser. By coding with JavaScript to listen to this DOM event, the measurement process is able to be executed automatically with certain accuracy.

In consideration of accuracy of measurement results and the convenience of automatic operation, the latter practice is selected as a fundamental part of the design and implementation in the experimental work of the study.

### **3.3. Design**

According to the results generated from the requirement analysis, possible solutions of the measurement model and alternatives are presented and discussed during design before starting the implementation. And the user interface of the measurement application is also drafted and discussed in this step of the research process.

#### **3.3.1. Measurement modules**

Since the choice to execute the measurement in automatic way has been made, [Section 3.2.4] the HTML DOM event, “onload”, which plays a key role to listen and handle the beginning and ending time moments of a page loading procedure, gets to be critical for the implementation in the measurement practice. In order to capture this event automatically during a set of referenced web pages, each web page used in the page loading process needs to be embedded into a sub frame of a scripted, unified web page which is



responsible for capture the DOM event, switching web pages, and committing detected results to database.

Based on the design to arrange necessary modular components around this multi-functional web page which supposed to be the main control point of the measurement environment, the fundamental components of the measurement environment have been identified, which consist of a set of scripted online pages as the access and control point, a storage module to provide pre-defined web pages as the measurement reference, and a database server module to store and fetch the generated data as the measurement results.

[Figure 3-4]

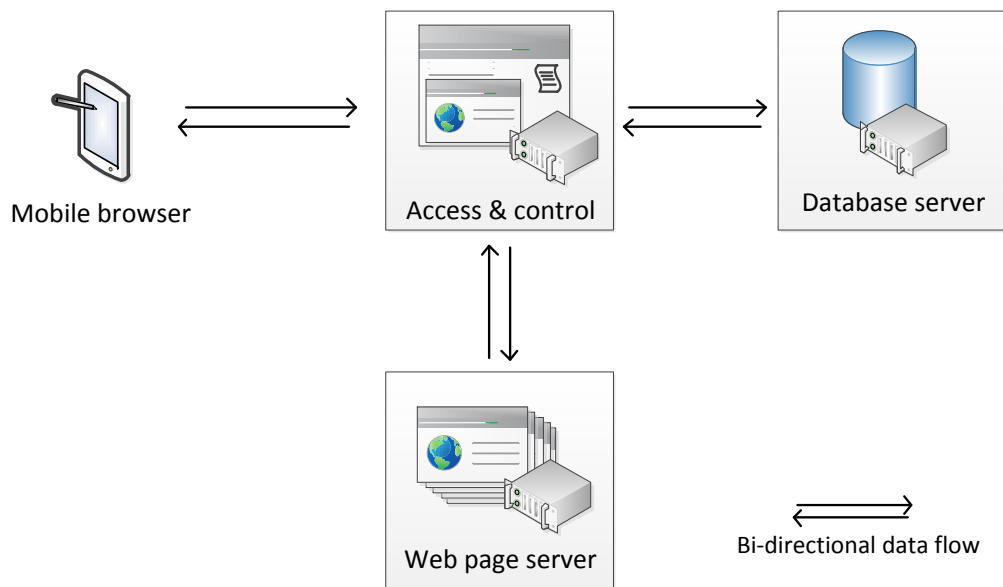


Figure 3-4: Modular components of automatic measurement environment

As the figure above illustrated, the access and control module of the measurement environment is the main entry for the measuring mobile browser to access the whole measurement environment. This module is responsible to invoke and switch the web page which will be allocated into a sub frame of the main page and launch the page loading process. When the embedded web page is loaded into the sub frame completely, the DOM event “onload” will be captured by the JavaScript event listener located in the main page. The time spent on page loading will then be calculated and recorded into the database module.

The three modular components presented in the figure are unnecessary to be departed. All of them can be set up to one single computer or nay other computing unit which have storage space. Additionally, except the access and control module which needs to provide network serving service as the entry for mobile device to access, the web page server and database server are even unnecessary to be real serving applications. It would be fine for the measurement environment to work as expectation if those web pages are not served but simply stored in storage. And the database server is also replaceable by other data logging approaches, such as XML file logging.

### **3.3.2. User interface of measurement control**

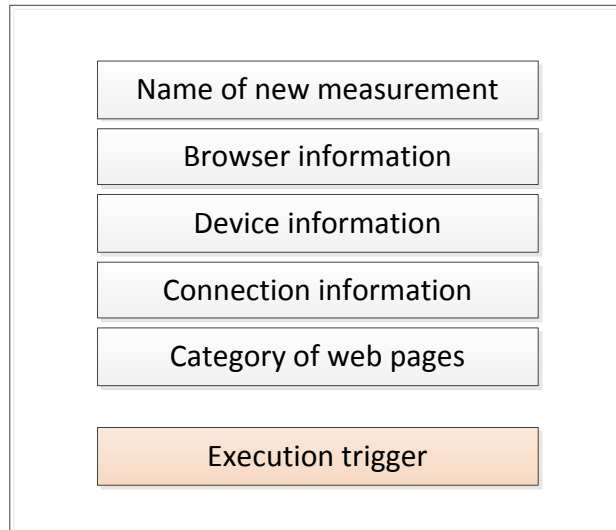
Based on decided measurement modules, [Section 3.3.1] the online entry and main control point module should be the only access for a mobile device to connect to the measurement environment. The user interface of the experimental implementation is therefore drafted here according to the functionality identified in the requirement analysis phrase [Section 3.2.1] of software engineering process.

#### **View of new execution**

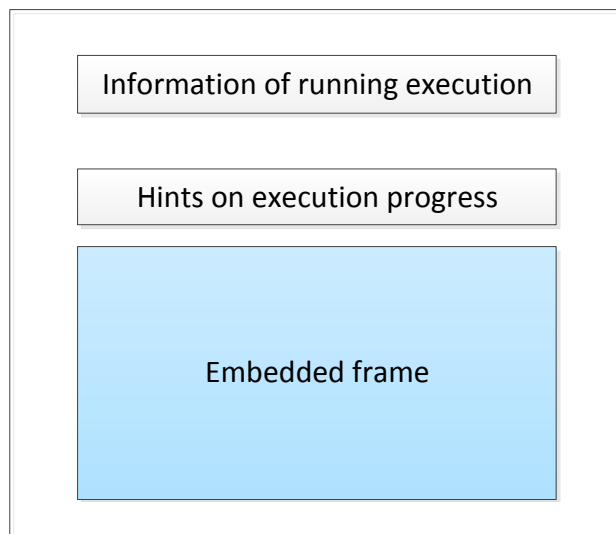
The view of new execution [**Figure 3-5**] is drafted in purpose to provide enough information for the control point to register the execution to database. Although there is some information easy to be acknowledged automatically by server, such as the execution time, there is some other information hard to be fetched from server side, which includes the measuring device model, the connection the device is using, and the category that the new measurement round would like to run on. These pieces of information have to be manually input by the user before executing the measurement.

#### **View of running execution**

The view of running execution [**Figure 3-6**] is prepared for the execution process of the measurement. The actual measurement happens in this page by loading web pages into the embedded frame of main page frame. After a web page is fully loaded, the controller will switch to load a second web page automatically. The progress will show at the top of the sub frame. After all assigned web pages being loaded, the browser should jump back to the main entry page immediately.



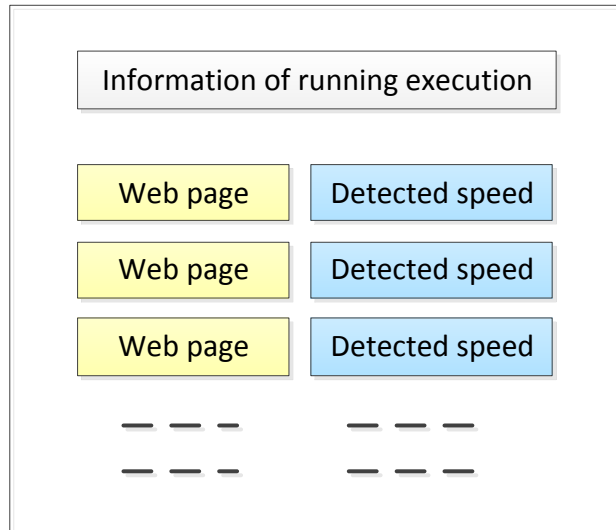
**Figure 3-5:** UI draft on view of new execution



**Figure 3-6:** UI draft on view of running execution

### **View of execution results**

The view of execution results [Figure 3-7] is drafted to present the detected page loading speed of each web page contained in the selected category. The speed of each web page is presented in the form of integer with time unit of millisecond.



**Figure 3-7:** UI draft on view of execution results

### 3.3.3. Possible measurement models

Based on acknowledged modular components of the automatic measurement environment, [Figure 3-4] it is easy to identify possible solutions to set up a measurement model which interprets the measurement operational process.

In the automatic measurement environment, the mobile browser in measurement needs to execute the page loading operation on a category of web pages which are located in the module of web page server. The location of the web page server module is therefore a critical factor which affects the time consumed on data transmission from web page server to the mobile device. This time consumption may influence the measurement results of the page loading speed.

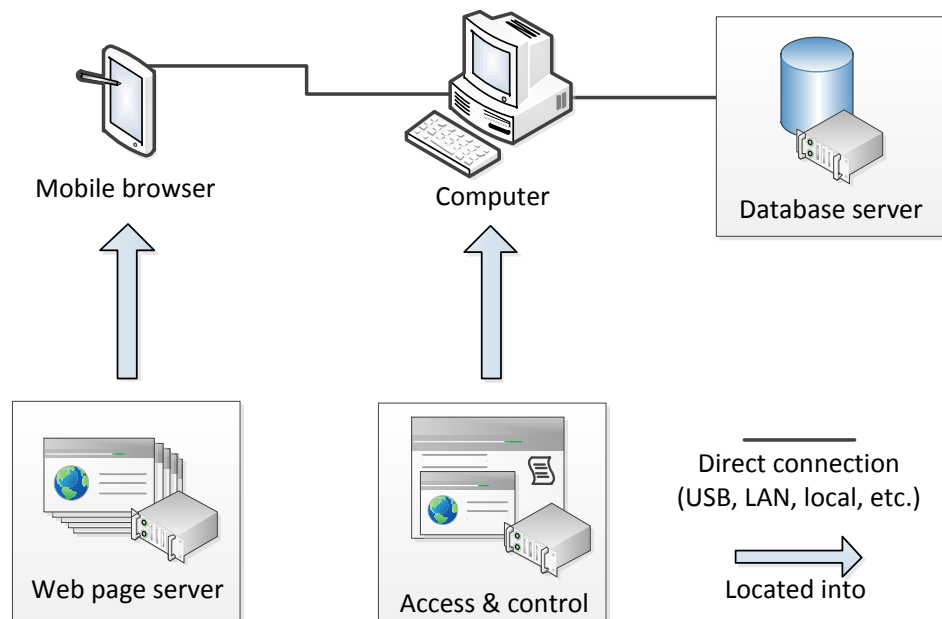
From technical points of view, there are several possible locations to address the module of web page server. Each selection of locations thus evolves to a sketch map of a measurement model. These models will be outlined in this section one after another.

#### Web page on device

The best way to eliminate the time consumption on transferring data from web page server module to measuring mobile device is to allocate the module onto the device. The time used for the mobile browser to request and fetch the content of pre-defined web pages can be mostly ignored in this way, since the only delay of time would happen based on the flash

storage IO (Input and Output) speed of the mobile device, which would be hundreds of thousands time faster than network transmission. [Figure 3-8]

The according measurement model contains a powerful computer to serve the entry and the main control point of the measurement environment. The computer and the mobile device in measurement are connected with wired data connection, which is commonly a USB (Universal Serial Bus) cable or an extensional dock of device indirectly.



**Figure 3-8:** A measurement model, web page serving on device

The wired connection between the mobile device and the computer can not only transmit the measurement data back and forward between the browser and the control point, but also provide a channel for the user working on the computer to control the behavior of the mobile browser and debugging the application remotely. And the all-around wired connection in the closed measurement environment gets rid of possible interference from unknown data traffic, which increases the accuracy of the measurement results greatly.

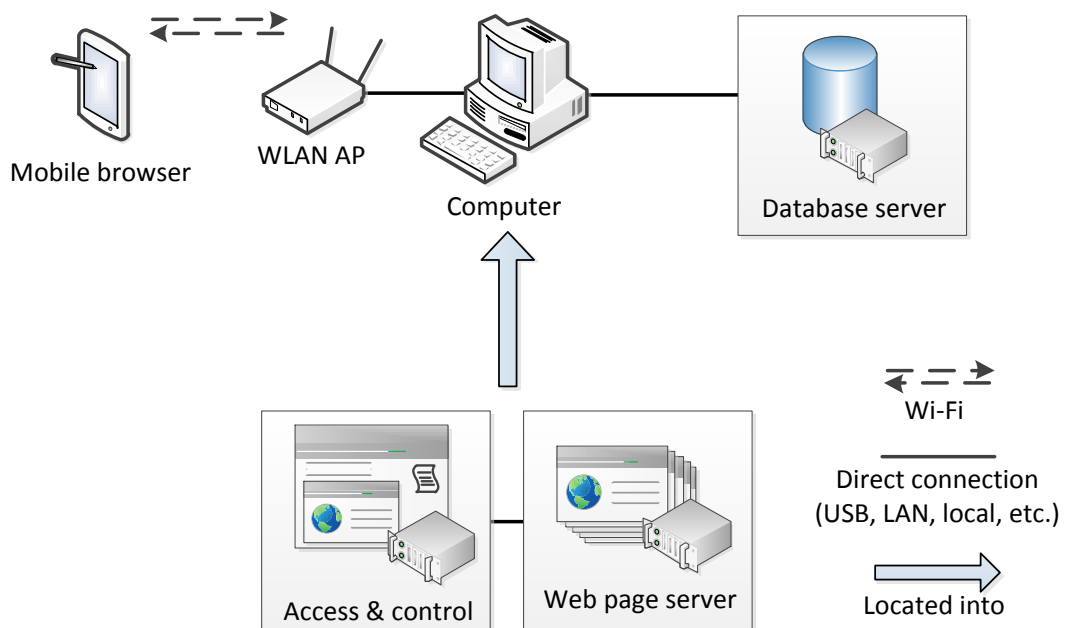
However, the establishment of such a measurement environment takes money and time. Since the hardware and software divergence, mobile devices from various manufacturers require completely different methods and data cables to connect and communicate to a computer. Some manufacturers block the possibility to manipulate their devices remotely

without broken the system. And the later configuration on the computer and the device is even more complicated.

Another drawback of this measurement model comes from the storage capacity of mobile device. In productive occasions it is common to pre-define a large category of web pages for the measurement on the page loading speed. Sometimes the storage space of the mobile device is overwhelmed by the content of these web pages, which prevents the use of such a kind of measurement model.

### Web page in local environment

Similar as placing the web page server module on mobile device, the measurement model which allocates the module into the computer together with access and control point gets benefit by closed environment to avoid interferences from outside network traffic. [Figure 3-9]



**Figure 3-9:** A measurement model, web page serving in local environment

Instead of strict wired connections, in this measurement model the connection between the mobile device and the computer is Wi-Fi, which is also recognized as WLAN (Wireless Local Area Network), by transmission through a WLAN access point directly connected to the computer. This practice removes the requirement to set up or configure the hardware and software environment of device. Instead, the measurement operation on the mobile

device turns to be the same as the steps to surf the Internet, which eliminates the inconvenience from diversity of setting up mobile device environment.

The drawback of this measurement model is also similar to the previous one. This measurement environment requires investment at the beginning on hardware and time, which would be spent to set up and configure the network interfaces.

### Web page distributed

Instead of other measurement models which centralized the pre-defined web pages into a web page server module, the measurement model without web page server module tries to provide the similar network connections as the real Internet surfing experience. In this measurement model, the control point located in the computer maintains a list of actual web addresses of referenced web pages. During the measurement operational process the mobile browser sends requests actually to the real web sites on the Internet, and gets responses with real-time web content back. [Figure 3-10]

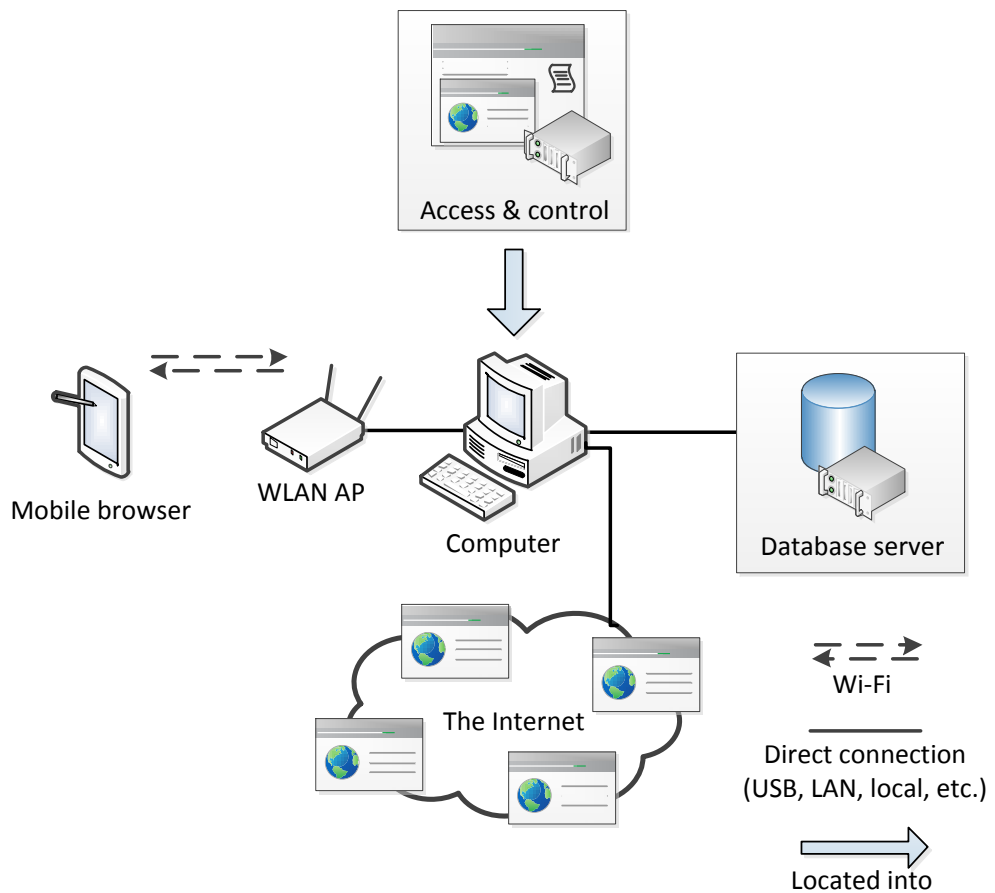


Figure 3-10: A measurement model, web page serving distributed

Using URL (Uniform Resource Locator) on the Internet to serve web pages as reference for measuring the page loading speed of a mobile browser avoids the creation and configuration of a web page server module. Instead, such a measurement model would guide the mobile browser to directly communicate to web sites serving on the Internet.

The removal of web page server module from a measurement model reduces the workload to collect the copies of pre-defined web pages, however introduces big uncertainty of time consumption on network connection which is heavily affected by the Internet traffic. The measurement results by using such a kind of measurement models are not consistent to be used as reference for further evaluation.

### Web page in cloud

With the help from more and more popular cloud computing services, it is nowadays more convenient to move network services into the cloud. In a measurement model with cloud computing platform, it is possible to allocate all fundamental modular components into the cloud, which eliminates the necessity of the local computing resources in the model.

[Figure 3-11]

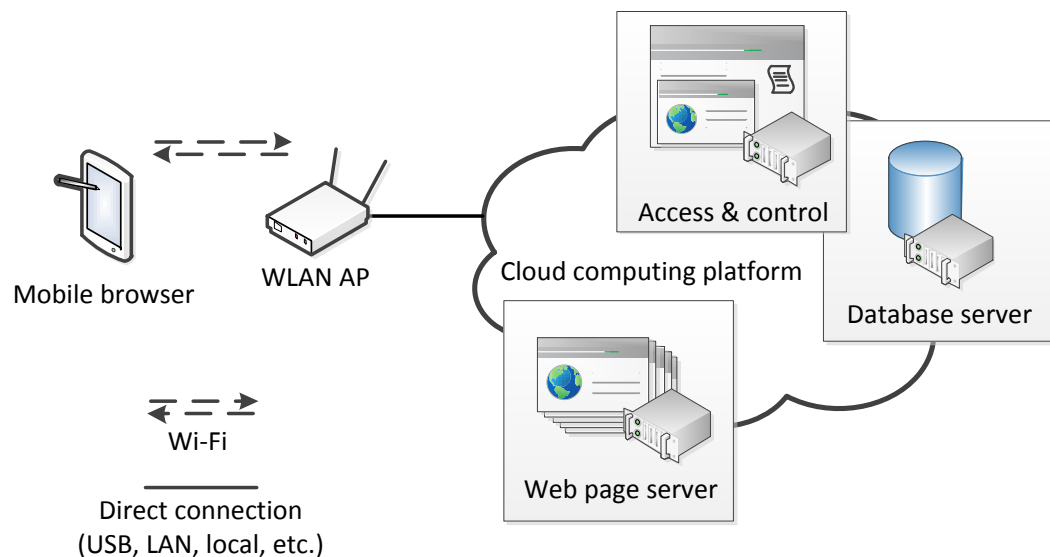


Figure 3-11: A measurement model, web page serving in cloud



Instead of forwarding the browser requests on web pages directly to the Internet, this model still contains the web page server module, which means all pre-defined web pages will be copied and collected to a centralized location on the cloud computing platform. The network connection between mobile device and the web page server module is not as stable as those in a closed measurement environment, but much better than distributed objects, and improvable with efforts from the cloud computing service provider.

### **3.4. Implementation**

The practical implementation work started after the user interface had been drafted out and the decision of development tools had been made. Unavoidably, there was some time consumed for the pre-study of necessary knowledge to become familiar with develop tools, which included the jQuery JavaScript library, webapp framework and database API (Application Programming Interface) provided by App Engine, and also the JavaScript and Python languages for coding practice. Fortunately, there were rich documents available online for web developers to learn these skills relevantly rapidly.

#### **3.4.1. Choice of hosting platform**

The choice of hosting location is critical to every web service and online application, which demands on being accessible from anywhere, and stable enough to be usable at any time. Depending on the programmable characteristic of the online application in this experimental part, the hosting platform needs to provide background database support, and an environment for running dynamic language scripts also.

The limitation of a non-existent budget [Section 3.2.3] required that the selected web hosting platform needs to be completely free of charge for its services. Unfortunately, the majority of hosting services available on the Internet did not satisfy this requirement when considering dynamic language programming ability.

The main alternative, however, is the public online hosting service provided by Google App Engine. [Section 2.4.2] As a feature-rich hosting platform, App Engine provides industrial scalability and is free of charge when the storage and page views per month do not exceed limited values. (Google, 2010) After balancing these values with the required server capability discussed in the requirement analysis [Section 3.2], it has been agreed that the limited values for the free usage of App Engine service is sufficient for the experimental part of this study.

Compared with a traditional web server solution such as the open source software bundle LAMP (Linux, Apache, MySQL and PHP), Google App Engine has better accessibility which allows anyone who wishes to help with the maintenance work of the experimental online application. Furthermore, App Engine provides relatively complete documentation of integrated APIs and a feature-rich SDK for web developers, which was absent from traditional LAMP-based web servers.

After the measurement of benefits and limitations among possible solutions, Google App Engine was finally chosen to be the hosting platform of the experimental online measurement application. As a matter of the fact, this decision influenced also the choice of online application development tools.

### **3.4.2. Selection of development tools**

In order to create the measurement application as verification to the indicated measurement process in research, the experimental part of the study will use a web application framework as a part of the development tools together with a JavaScript library which contains the necessary Ajax functionalities. All choices of development tools highlighted reducing the programming workload while minimizing the necessary time required learning how to use and to maintain the implemented work.

#### **Web application framework**

Based on the decision of using the cloud computing services provided by Google, its default web application framework integrated in Google App Engine, webapp, turned out to be the first, as well as the best, choice for the experimental part of this study. Webapp is simple but powerful enough to be used in the research work as it provides all general features a web application framework should have. Since it is built-in to App Engine by default, it is not necessary for developers to be concerned about the binding work which is compulsory for any other web application framework to work with the App Engine platform.

#### **JavaScript library**

The JavaScript library is commonly used nowadays in web development on the Internet. It releases web developers from tedious JavaScript coding and debugging, and make it possible for beginners to implement the comprehensive behaviors of web page elements. Although webapp in Google App Engine does not contain a JavaScript library by default, there are many JavaScript libraries available with a free software license. As most of those libraries provide similar functionalities, one would be enough to be used in the research work.

Based on the survey of usage of common JavaScript libraries, jQuery is the most popular JavaScript library used on the Internet. (BuiltWith, 2010) It was also chosen to be used for the experimental part of the research.

### **Programming language**

Based on the decision of using Google App Engine as hosting platform and webapp as the web application framework, Python turns out to be the ideal choice of working language for the programming part of the research. It is a remarkably powerful dynamic programming language that is used in a wide variety of application domains and available for all major operating systems. (Python Software Foundation, 1990-2010)

#### **3.4.3. Capacity of measurement model**

In order to enhance the accuracy of detected data of the page loading speed of a web browser, in some productive environment of mobile device manufacturers it is common to pre-define a large category of web pages as the measurement reference, which may consist of hundreds of home pages of popular web sites worldwide. The overall content as composition of these web pages might be too much to be stored or handled in the practical application implemented in the experimental part of the study.

As the main purpose of the experimental work is to prove and verify the measurement process summarized at the end of the research, it seems unnecessary to provide capacity large enough for business-level data storage and manipulation. Instead, a small category of web pages and their content should be enough to commit to the measurement process as the reference of the page loading speed. These web pages should be collected from the most popular web sites which have the highest web traffic from visitors on the Internet.

### **3.5. Verification**

In the Waterfall model of software engineering process, [Figure 3-1] verification commonly happens after the implementation work has finished according to the design. In agile style programming, [Figure 3-2] the verification actually happens together with design and implementation from time to time. It is quite difficult to specify in which time period the work of verification separated from design or implementation.

As many tiny issues, for instance coding syntax errors, were found actually during implementation and solved easily, they will not be mentioned here. Instead, the issues found during integration testing which were so serious that the whole structure of the

research work had to be considered once again because of those are discussed in the next sections.

### **3.5.1. Browser compatibility**

Compatibility testing is a critical step for web application development processes because of the different Rendering engines among various browser applications. [Section 2.1.2] As browser applications are software products produced by different commercial companies, each company implements web browser standards in an independent way. The browser's parser of CSS is one example of these conflicts.

CSS (Cascading Style Sheets) is commonly used on web sites as a basic standard to regulate presenting styles of text and pictures in web pages. W3C (W3C, 2010) has published the CSS standards from version 1 to 3, to reflect the demand of web design requirement. (W3C, 2010) However, all browsers do not follow these de facto standards. Some grammars were replaced by their unique implementation out of standard. Some features were even not implemented at all.

### **3.5.2. Category of mirrored web pages**

In order to verify the implemented work in the experimental part of the study, the collection of web sites duplicated locally as mirroring components in the experimental part of the study are presented in a unified form. The rank orders of these web sites are as the same as those of the Global Top Sites being presented on the website of Alexa.com, which is freely available on the Internet. [Figure 3-12]

Because of the issues which are found out during the experimental part of the study, some web sites listed in the form below could not be correctly mirrored in order to fetch correct results during the measurement. Therefore these sites are marked here with different background colors for classified reasons. Without counting in these problematic web sites, the total number of available sites in the form is 20, which is the maximum number of mirrored web sites being manipulated in the experimental part of the study. [Section 3.4.3]

| Rank | Site Name           | Web Address    | Brief Description                       |
|------|---------------------|----------------|---|
| 1    | Google              | google.com     | Internet search engine                  |
| 2    | Facebook            | facebook.com   | Social networking service               |
| 3    | YouTube             | youtube.com    | Worldwide video sharing service         |
| 4    | Yahoo!              | yahoo.com      | Internet portal and service provider    |
| 5    | Blogger             | blogspot.com   | Weblog publishing tool                  |
| 6    | Baidu (百度)          | baidu.com      | Search engine in Chinese language       |
| 7    | Wikipedia           | wikipedia.org  | A free encyclopedia on wiki             |
| 8    | Windows Live        | live.com       | Search engine from Microsoft            |
| 9    | Twitter             | twitter.com    | Social networking and microblogging     |
| 10   | QQ                  | qq.com         | Internet service portal in China        |
| 11   | MSN                 | msn.com        | Internet service portal                 |
| 12   | Yahoo! Japan        | yahoo.co.jp    | Japanese version of Yahoo!              |
| 13   | Sina (新浪)           | sina.com.cn    | Internet news in Chinese language       |
| 14   | Google India        | google.co.in   | Indian version of Google                |
| 15   | Taobao (淘宝)         | taobao.com     | Person to person transaction in China   |
| 16   | Amazon              | amazon.com     | Customer-centric online purchase        |
| 17   | LinkedIn            | linkedin.com   | Networking focusing on business         |
| 18   | Google Hongkong     | google.com.hk  | Chinese version of Google               |
| 19   | WordPress           | wordpress.com  | Free blog management                    |
| 20   | Google Germany      | google.de      | German version of Google                |
| 21   | Bing                | bing.com       | Search engine from Microsoft            |
| 22   | Google UK           | google.co.uk   | UK version of Google                    |
| 23   | Yandex (Яндекс)     | yandex.ru      | Search engine in Russian language       |
| 24   | eBay                | ebay.com       | Person to person auction                |
| 25   | Google Japan        | google.co.jp   | Japanese version of Google              |
| 26   | Microsoft           | microsoft.com  | Main site of Microsoft corporation      |
| 27   | Google France       | google.fr      | French version of Google                |
| 28   | NetEase (网易)        | 163.com        | Internet portal and networking in China |
| 29   | Google Brasil       | google.com.br  | Brazilian version of Google             |
| 30   | Google User Content | (N/A)          | User content of Google                  |
| 31   | Flickr              | flickr.com     | Free picture galleries                  |
| 32   | FC2                 | fc2.com        | Video sharing service in Japan          |
| 33   | Mail.ru             | mail.ru        | Email service provider in Russia        |
| 34   | Google Italy        | google.it      | Italian version of Google               |
| 35   | Craigslist          | craigslist.org | Centralized community network           |

Explanation of the background colors to web sites in rows:

|        |
|--------|
| Normal |
| Green  |
| Red    |
| Yellow |

Available in the experimental part  
Duplicate site content  
Fail to execute page loading for evaluation  
Unable to mirror locally

Figure 3-12: Top 35 of Global Top Sites retrieved from Alexa.com

## 3.6. Deployment and Maintenance

The deployment step in software engineering means to deliver the finished coding results into the real working environment. For web applications, this means setting up the server-side services and getting those ready to serve client users. Maintenance is the last step of the software engineering process. It commonly consists of fixing bugs found after deployment, adding new features onto existing applications, and other code modification to fit the user's requirement changes.

### 3.6.1. Deployment

In web application development, deployment could be more difficult than local software applications since web service has to be a combination of services by the HTTP server, database server, language runtime, and other relative services. The configuration and adjustment of all these interconnected services are time-consuming and error-prone.

Fortunately, Google App Engine provides a powerful SDK (Software Development Kit) for web developers to deploy the tested application onto the remote server. The SDK is responsible for detection of any file changes since the last time when new site content was submitted, and uploads the changed files automatically for the user.

Google App Engine supports also debugging on the remote server directly. After the coding work has been deployed onto the App Engine platform, App Engine will record all warnings and errors happening during the serving time, which are easy to access by web developers. This helps web developers to easily trace hidden issues happening only in the real environment.

### 3.6.2. Maintenance

Because the experimental part of the study aimed to create an online measurement application to measure the page loading speed of mobile browsers, maintenance was certainly important for following the feedback after tool usage and upgrading the code and functionalities.

In order to help other developers to become familiar with the code and the whole structure of the experimental part of the study as quickly as possible so that they could continue the maintenance work later after this study, all parts of the research work have been saved in the source code and uploaded to a public online version control system. The usage of a version control system helps several developers to share source code changes and prevent the risk of data loss or undesired operations resulting from human mistakes.

## 4. Results

In this chapter, the results of the research are presented here which consist of the latest user interface of the online measurement application and the software programming practice during the experimental work. In order to prove the usability of the implementation, several mobile devices and integrated mobile browsers were also used to verify the generated measurement process and its actual operation practice. For the coding presentation, the results of the research of conflicts among Google App Engine platform, jQuery, and related coding guidelines are presented and discussed.

### 4.1. Decision of Design

This section described final decisions made during the design phrase of the measurement process, which consist of the conclusion of the measurement model, the optimized operational process, and adopted user interface of the measurement application.

#### 4.1.1. Measurement model

During the design phrase of the research process, the fundamental modules necessary to establish a workable measurement environment have been identified and discussed. [Section 3.3.1] And then the possible locations of the web page server module are pointed out and according measurement models are illustrated and discussed on the advantages and disadvantages. [Section 3.3.3] Based on all these contexts and the output of the requirement analysis phrase in the research process, [Section 3.2] the decision of the measurement model has been made.

In the experimental part of the study the practical implementation of the measurement model and the operational process has decided to adopt the measurement model which located the web page server module in cloud. **[Figure 3-11]**

The three fundamental modular components of the measurement environment, which consist of the access and control module, the web page server module and the database server module, are all implemented into the cloud computing services provided by Google. By serving the application access and control point online, the implementation of the measurement environment ensures its availability worldwide all the time. The centralized allocation of web pages which are categorized as reference of measurement results decreased the difference of data transmission distance on network between the mobile browser and the loading web page. And by using the database query APIs integrated in

cloud computing platform, the manipulation of measurement data gets much easier than operating offline database system, which would cost money and more time to set up.

The access point which connecting mobile devices and the cloud can be any single wireless access point reachable without access restriction. When the measurement rounds would be executed on several different mobile devices, it is a good practice to keep these devices at the same place when setting up the WLAN (Wireless Local Area Network) connection and executing the measurement processes. The measurement should be executed in a relatively closed location, such as a room without other people who might use wireless devices, in order to mostly decrease interferences from other connection requests to the access point.

#### **4.1.2. Measurement process**

The measurement practice for one mobile browser is as simple as looking through a web site on the Internet. Using the mobile browser to visit the web address which will lead the user to the home page of the online application hosting on the cloud computing platform, which is also the entry and main control point of the measurement environment, the table of pre-setting options for execution and the execution button will appear in the browser's window. After pressing the execution button, the browser will be redirected to the running page which starts the execution automatically. The progress of the execution will be shown and updated all the time during the execution. At the end of the execution, the running page will redirect the browser back to the main page after the last web page fully loaded, since the access links to review measurement results are located beneath the execution button. The measurement results are reserved in the database in cloud which means they are accessible from anywhere at any time by any network device.

#### **4.1.3. User interface**

During the design phase there are three main views of the main entry and control point of the measurement application being drafted out, which consist of view of new execution, view of running execution and view of execution. [Section 3.3.2] Based on these drafts, the final implementation of the application user interface is organized into three HTML pages, the main page, the run page and the result page.

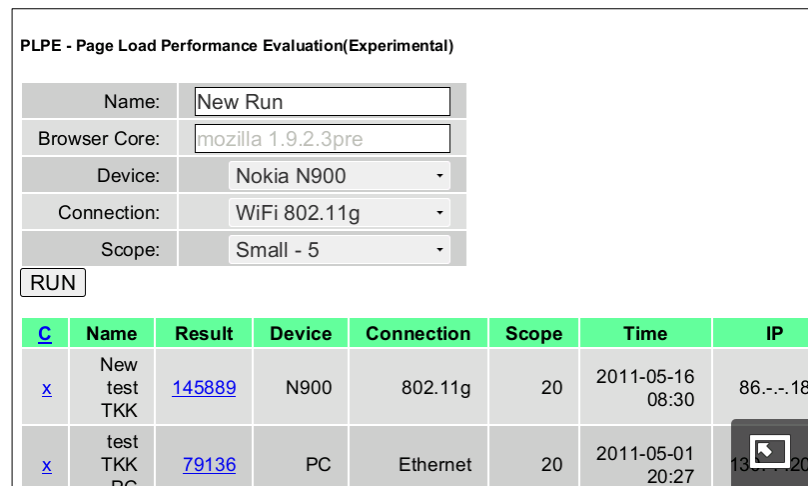
The final implementation of these pages are presented here by taking the screenshots on Nokia N900 mobile device in order to verify the usability of the user interface on mobile device.



## Main page

The main page shows the configuration options which are helpful to recognize the execution results after running a round of the page loading speed measurement. [Figure 4-1] As the design phrase pointed out, there are several options not able to be detected by web application automatically so that the user has to manually input each of them to leave correct information to identify the measurement results in the future. [Section 3.3.2]

In addition to the table of execution options, in this page a summary list of all executed measurement rounds is presented also for user to review the measurement results conveniently. At the most left side delete buttons are provided also in order to remove useless results.

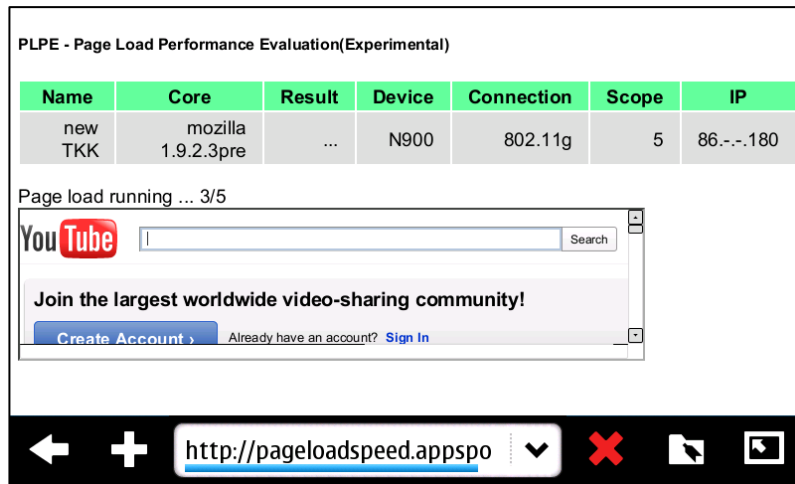


| PLPE - Page Load Performance Evaluation(Experimental) |                    |        |        |            |       |                  |           |
|---|--------------------|--------|--------|------------|-------|------------------|-----------|
| Name:   | New Run            |        |        |            |       |                  |           |
| Browser Core:   | mozilla 1.9.2.3pre |        |        |            |       |                  |           |
| Device:   | Nokia N900         |        |        |            |       |                  |           |
| Connection:   | WiFi 802.11g       |        |        |            |       |                  |           |
| Scope:  | Small - 5          |        |        |            |       |                  |           |
| RUN   |                    |        |        |            |       |                  |           |
| C   | Name               | Result | Device | Connection | Scope | Time             | IP        |
| x   | New test TKK       | 145889 | N900   | 802.11g    | 20    | 2011-05-16 08:30 | 86.-.-.18 |
| x   | test TKK PC        | 79136  | PC     | Ethernet   | 20    | 2011-05-01 20:27 | 13.-.-.20 |

Figure 4-1: Screenshot of Main page, captured by using Nokia N900

## Run page

The run page is responsible to automatically execute the actual measurement by loading assigned web pages into the sub frame of the web page. [Figure 4-2] The automatic behaviors are performed by the embedded scripts written in JavaScript language with help from jQuery JavaScript library. A hint line above the sub frame indicates the page loading progress in real time. After all assigned web pages have been loaded, the measuring browser jumps back to the main page with the latest results showing on the result list.



**Figure 4-2:** Screenshot of Run page, captured by using Nokia N900

### Result page

The result page shows the entire page loading speed detected in one measurement round. The values are presented as integers with unit of millisecond, which is the smallest unit of time able to manipulate in JavaScript language. [Figure 4-3]

The referenced web pages are listed in order of the global web traffic, which was collected by Alexa.com service. [Section 3.5.2]

| Name               | Core                  | Result | Device | Connection | Scope | IP        |
|--------------------|-----------------------|--------|--------|------------|-------|-----------|
| New<br>test<br>TKK | mozilla<br>1.9.2.3pre | 145889 | N900   | 802.11g    | 20    | 86.--.180 |

| Rank | Website   | Page load speed |
|------|-----------|-----------------|
| 1    | google    | 4871            |
| 2    | facebook  | 3922            |
| 3    | youtube   | 8168            |
| 4    | yahoo     | 12170           |
| 5    | blogger   | 8381            |
| 6    | baidu     | 3624            |
| 7    | wikipedia | 4347            |
| 8    | twitter   | 10191           |

**Figure 4-3:** Screenshot of Result page, captured by using Nokia N900

## 4.2. Measurement practices

Verification on the online measurement application which is implemented in the experimental part of the study is important to prove the usability of the generated results from design. In order to simulate the evaluation process, several mobile browsers integrated in separate devices from various device manufacturers participate in verification.

### 4.2.1. Measurement participants

Several mobile devices are selected to join the verification for the measurement on the page loading speed of their integrated mobile browsers. Every mobile device has unique hardware capacities and software stacks which essentially affect the measurement results of the page loading speed of their mobile browsers. [Figure 4-4]

| Browser        | Safari                            | MicroB        | Grob          | Firefox        |
|----------------|-----------------------------------|---------------|---------------|----------------|
| Device         | Apple iPod touch (1st generation) | Nokia N900    | Nokia N950    | Desktop PC     |
| CPU power      | 400 MHz*                          | 600 MHz       | 1 GHz         | 2.66 GHz       |
| DRAM size      | 128 MB                            | 256 MB        | 512 MB        | 4 GB           |
| Connectivity** | Wi-Fi 802.11g                     | Wi-Fi 802.11g | Wi-Fi 802.11g | Ethernet       |
| Release date   | Sep, 2007                         | Sep, 2009     | Jun, 2011     | N/A            |
| OS             | iOS 3.1.3                         | Maemo 5       | Meego         | Windows 7      |
| Browser engine | WebKit 528.18                     | Gecko 1.9.2.3 | WebKit 531.4  | Gecko 1.9.2.16 |

\* The CPU power was under-locked from 620 MHz by the device manufacturer.

\*\* 802.11b is supported by all but not in use. 802.11n is supported by Nokia N950 but not by the Wi-Fi access point available in measurement.

**Figure 4-4:** Mobile browsers within devices in measurement

To provide a reference for the measurement results, a web browser application running on a desktop computer is introduced into the verification process. Since the implementation of the online measurement tool aims to be portable on various hardware and software platforms, the measurement execution to the web browser application on the desktop computer has no obvious difference against mobile browsers.

#### 4.2.2. Verification across browsers

Before the execution of measurement on the page loading speed of mobile browsers, all of the mobile devices have been upgraded to be ready with the latest version of official firmware, which includes the latest version of mobile browsers as well. In order to get rid of the influence coming from browser cache on device to the detected page loading speed, [Section 2.1.2] the browser cache, browsing history records and saved cookies were all cleaned up right before each time to start the execution. For every mobile browser and its device, the measurement was executed for three times to provide convincing results in purpose of verification. [Figure 4-5]

|         | Result 1 | Result 2 | Result 3 |
|---------|----------|----------|----------|
| Safari  | 354362   | 362378   | 356527   |
| MicroB  | 145889   | 147379   | 136680   |
| Grob    | 104253   | 108677   | 112321   |
| Firefox | 41753    | 39068    | 41122    |

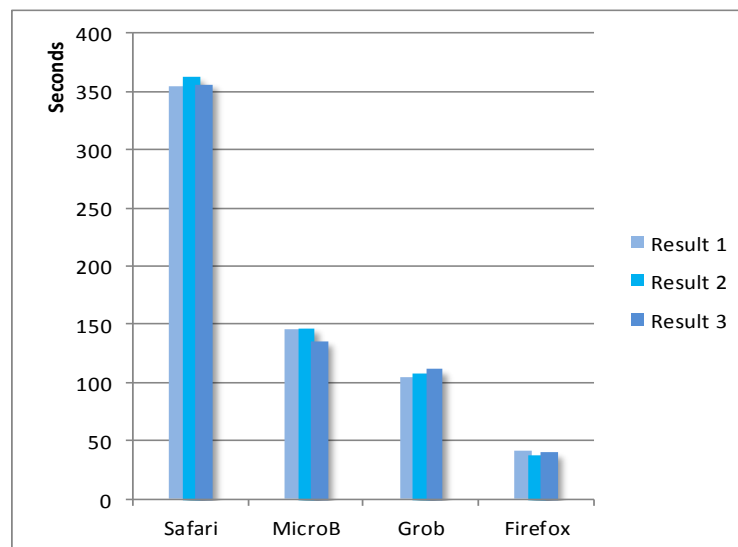


Figure 4-5: Page loading speed across browsers

From the measurement results, it is obvious to notice the critical influence from the hardware capacities to the page loading speed of mobile browsers. The more powerful CPU and larger memory, the less time spent on loading the category of web pages [Section 3.5.2] by the mobile browser.

Although detected values of all three measurement results are not equal for each mobile device, they are close to the average, which was relatively stable to indicate the page load performance of the mobile browser within its contexts. The stability of the measurement results also verified the implementation of the measurement process.

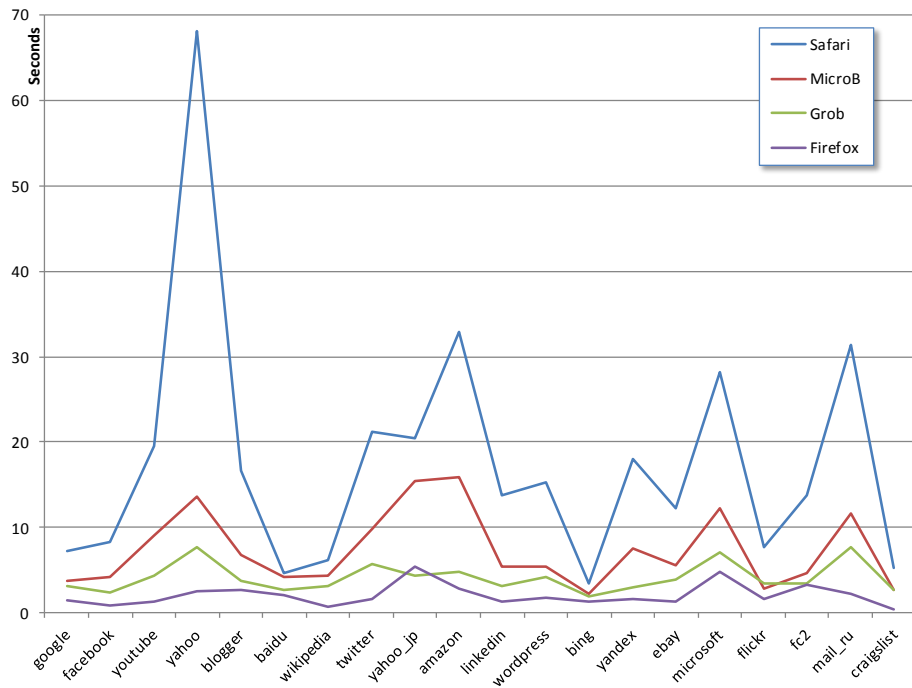
#### 4.2.3. Verification across web pages

For each web page mirrored in the web page server module as the reference for the measurement, the page loading speed in a mobile browser may reflect the data size of the content which is needed to be fetched and rendered by the browser. The results are therefore beneficial for web site developers to find out reasons of latent delays and do optimization on proper targets.

On the other hand, the detailed measurement results for each web page across mobile browsers may be an evidence to prove the stability of the measurement environment and operational process by looking through the consistency of the detected values across measuring mobile browsers. [Figure 4-6]

In the table below, all presented values of the page loading speed of a mobile browser are the average of results from three rounds of the measurement.

|            | Safari | MicroB | Grob  | Firefox |
|------------|--------|--------|-------|---------|
| google     | 7215   | 3721   | 3210  | 1411    |
| facebook   | 8302   | 4142   | 2412  | 819     |
| youtube    | 19489  | 9058   | 4286  | 1324    |
| yahoo      | 68066  | 13586  | 7652  | 2541    |
| blogger    | 16647  | 6740   | 3688  | 2624    |
| baidu      | 4712   | 4216   | 2724  | 2076    |
| wikipedia  | 6256   | 4331   | 3190  | 724     |
| twitter    | 21276  | 9793   | 5653  | 1638    |
| yahoo_jp   | 20491  | 15509  | 4418  | 5355    |
| amazon     | 32861  | 15861  | 4816  | 2781    |
| linkedin   | 13712  | 5379   | 3194  | 1362    |
| wordpress  | 15236  | 5490   | 4192  | 1834    |
| bing       | 3420   | 2246   | 1886  | 1323    |
| yandex     | 18017  | 7564   | 3021  | 1637    |
| ebay       | 12226  | 5637   | 3857  | 1319    |
| microsoft  | 28252  | 12296  | 7158  | 4741    |
| flickr     | 7700   | 2885   | 3427  | 1656    |
| fc2        | 13778  | 4637   | 3466  | 3252    |
| mail_ru    | 31363  | 11575  | 7715  | 2270    |
| craigslist | 5343   | 2713   | 28712 | 435     |



**Figure 4-6:** Average page loading speed

From the graph, the consistency of the measurement results among different mobile browsers looks good. For each mirrored web page as a part of reference, the relative positions of the page loading speed across mobile browsers mostly keeps in order. A web page loading faster in one browser was also faster in another browser. And a slower web page kept slower no matter being loaded by which browser in the next round. Such a kind of consistency verified the usability of the online measurement application implemented in the experimental part of this study.

## 5. Discussion

In this chapter, an attempt will be made to answer the questions presented at the beginning of the study [Section 1.2] based on the research work and results of the experimental part of the study. And then the implemented online measurement application is benchmarked among several alternative performance measurement solutions available in public domain to specify its possible advantages and limitations, and suitable occasions of usage.

### 5.1. Answering Research Questions

In the beginning of this study, there were three research questions defined in the *Introduction* chapter [Section 1.2]. Based on the conducted research and the results presented in the *Research* chapter [Section 3] and the *Results* chapter [Section 4], the research questions will now be reviewed and discussed.

#### **1st Question: How to evaluate page load performance of mobile browsers accurately and efficiently?**

The page load performance of mobile browsers is evaluated commonly based on the measurement on the page loading speed, which depends on not only the hardware capacities of the mobile device, but also the software design and implementation in the browser application, the throughput and reliability of the wireless connection, and the content of the web page or the set of web pages used in the measurement process.

In order to measure and compare the page loading speed amongst mobile browsers, a set of web pages and their links are commonly prepared to be used as a reference, so that these mobile browsers operated in the measurement have similar workload when being requested to load the assigned suite of web pages.

In addition to avoid loading uncertain web content, such as the web contents randomly generated by dynamic scripts embedded in web pages, to keep consistent workload on page loading process of mobile browsers, there are also approaches to enhance the reliability of mobile device's network connection. A common solution used in the actual production environment is to download and mirror all remote content of referenced online web pages locally, which means storing and serving these web pages directly on the measuring mobile device. This on-device simulation for the targets of page loading requests eliminates all the possible uncertainty of network connection, but is relatively complicated to deploy to alternative mobile devices because of various hardware modules, software

stacks and system configuration. And it is even not possible to be adopted by some devices when the total data size of referenced web content exceeds the limited storage space available on device.

In the experimental part of the study, the implemented measurement process on page loading speed tries to solve this problem by downloading and serving the referenced web pages remotely from the measuring device, while providing consistent network connection for the mobile browser on the device to communicate with. This solution does not clean up all concerns about the reliability of wireless connection, but significantly reduced the uncertainty on time spent to access each web page by centralizing all web content together. And most importantly, by adopting this solution the complexity on setting up mobile devices is avoided, which increases the efficiency greatly.

Instead of gazing the screen of a mobile device with a stopwatch in hand, in the experimental part of the study the implemented online application determines the page loading speed of a mobile browser automatically with the help of Ajax techniques and JavaScript libraries. By reallocating the page loading operation to a sub frame of a web page and detecting a DOM event which occurs when all the content of the sub frame is finished loading to the browser's screen, the application is able to acknowledge the exact moment of time, accurate to millisecond, when the page loading is finished, and thus the spending time period after subtracting the initiating time recorded earlier. This software implementation enhanced the accuracy greatly against other awkward approaches, and reduced the amount of manual intervention.

## **2nd Question: What are the benefits and drawbacks of implementing the measurement model on a cloud computing platform?**

In the experimental part of the study the implemented measurement application is created in the online environment with the help of several modern web technologies, such as Ajax, JavaScript library, and a Web Application Framework, webapp (Google, 2008), which is integrated in the chosen cloud computing platform, Google App Engine.

It eliminates the requirement to invest, set up and maintain a specific environment for the execution process on the page loading speed measurement of mobile browsers. A traditional measurement environment commonly consists of hardware computing units, software operating systems and web serving applications, flexible wireless network connections, and human resources for necessary configuration, adjustment and maintenance. But by implementing and hosting the experimental work onto a cloud



computing platform, all these redundant physical and mental investment were avoided, thus increased the efficiency of the evaluation.

Allocating the measurement application into a cloud computing platform also simplified the process for mobile devices to access the entrance of the measurement environment. There were not additional network gateway settings or proxy server settings necessary to be preset up on the mobile device for using the mobile browser, since fetching data located on a cloud computing platform is as exactly the same as surfing the Internet.

Furthermore, keeping the experimental work on a cloud computing platform ensured the running instance of measurement application always up to date. Newly added features or code patches to fix known software bugs would be committed and deployed to the working environment at the first moment when available, which would then be accessible immediately by all measuring mobile devices.

In addition to all advantages of cloud computing services above, as the experimental part of the study selected Google App Engine as the cloud computing platform in the actual implementation, the workload on programming was significantly decreased with the help of comprehensive online programming APIs provided by App Engine and other infrastructures from Google.

On the other hand, adopting a cloud computing platform into the measurement environment as the hosting server for the referenced web pages for the page loading speed determination introduced uncertainty into the measurement results. Instead of running the web page server on mobile device which eliminates the affects from the reliability of network connection totally, or on a workstation server in a closed network environment, the wireless network connection between a measuring mobile device and the online server on a cloud computing platform might be relatively unreliable and easy to be affected by other possible network traffics running across. To decrease this kind of effects, a good practice of the measurement process is to access the online measurement application at the same physical location with the same Wi-Fi access point for different mobile browsers in order to achieve similar wireless connection bandwidth and possible network delay. The selection on Google App Engine which is running under Google's Internet infrastructures was also effectively helpful on decreasing this kind of negative effects as this Internet giant keeps doing its best to improve the access speed to its online services worldwide.

### **3rd Question: How to improve the page load performance of a mobile browser?**

As the research results pointed out during the study, the page load performance of a mobile browser depends on not only the hardware capacities of the measuring mobile device, but also execution performance and code optimization of software components of the browser structure. Additionally, the content of referenced web pages which are loaded by the mobile browser is also an important dependency if network connection maintains steadily during the measurement process.

The hardware capacities can be improved through introducing more powerful CPU, faster and more DRAM, and better production methods by mobile device manufacturers. The prices of these frontier digital gadgets will get lower and lower as time goes by, which would benefit the general performance of mobile browsers without more cost from users eventually.

The execution performance of the structural components of a mobile browser can be increased by browser developers. As the latest version number of a mobile browser gets higher and higher, the source code of the browser application is updated frequently. Some main components of a mobile browser, such as layout engine and JavaScript Interpreter, are being developed and maintained separately in open source societies outside of mobile device manufacturers. Such kinds of separation of functional duties and source sharing activities help global browser developers to locate and fix issues much quickly than years ago, and make it possible to do code refactoring and optimization much more frequently.

Another critical factor to decide the page loading speed of a mobile browser is the design of the web page which is located on a web server from the other side of the Internet. The less data needed to be transferred from a web server and a browser, the faster the page loading speed of that page. A lot of web service providers have learnt that and started optimizing the content contained in their home pages which are critical to access all the services they provide.

Traditional optimization practices for desktop web browsers are not suitable for mobile browsers, since the page load performance of mobile browser is related to not only the size of web content but also the behaviors of dynamic scripts and other embedded content which may initiate additional data requests. To balance the user experience on desktop environment and the performance of mobile browsers, a great number of web sites provide a second home page as the online entrance especially for mobile device users. Such a kind of practice has been proved to be successful in the market.

However the workload on re-engineering the home page of an online application might be huge with the expectation to make a unique web page layout comfortable to users of every mobile browser available in the market.

## 5.2. Benchmarks

In this section we compare the implemented measurement application with some other performance measurement tools which are freely available on the Internet for web browser developers to perform the evaluation across browsers. The comparison helps to find out the advantages and disadvantages of the design and implementation in the experimental part of the study, and provides a reference for possible future work for fixing issues and adding more features.

### 5.2.1. Comparison with SunSpider

SunSpider is a benchmark suite that aims to measure JavaScript performance on tasks relevant to the current and near future use of JavaScript in the real world, such as encryption and text manipulation. (SunSpider, 2011) There are a lot of browser developers worldwide who use SunSpider to compare the JavaScript performance of different browsers. The benchmark suite tries to focus on the kinds of actual problems developers solve with JavaScript today, and the problems they may want to tackle in the future as the language gets faster.

SunSpider is a JavaScript benchmark tool which tests the execution performance of a web browser on core JavaScript language only. It does not work when trying to test the DOM manipulation speed or other APIs of a browser. It is designed to compare different versions of the same browser, and different browsers to each other.

Compared with SunSpider, the implementation in the experimental part of the study is a measurement tool focusing on the page load performance of a browser, which related to not only the JavaScript execution performance but also performance of other structural components of a browser which affects the page loading speed.

However, the subsections of the benchmarks available to be measured by SunSpider are quite many, which consist of 3d rendering, variable access, bitwise operations, control flow constructs, code cryptography, date object operations, mathematical computations, regular expressions, and string processing. (SunSpider, 2011) It is a great feature to have such many specific measurement aspects for detailed evaluation, which the implementation in this research lacks of.

### **5.2.2. Comparison with Speed-battle**

Speed-battle is an online application which aims to measure the JavaScript speed of visitor's browser and shows other visitors' results with the same set of operating system and browser. (Gürgens, 2011) The application has a fresh and cool user interface which will show the measurement results immediately when a visitor tries to access the web site for the first time.

The measurement results generated from Speed-battle depend on different parameters, which include the performance of CPU, browser, browser plugins, operating system and system memory, and the number and kind of processes running in parallel during the measurement execution. According to the measurement results, the application will lead the visitor to a reference value for the performance of the browsing environment, which consists of both software and hardware.

Compared with Speed-battle, the implementation in the experimental part of the study is similar on providing tidy and friendly user interface to users. Both measurement applications are located on the Internet and provide convenient online entrance to launch the measurement execution. And both measurement processes focus on the performance in general performance of the platform instead of the browser only.

As the Speed-battle does not optimized specifically for mobile browsers, the user interface of the measurement application looks awkward and difficult to operate on compact screen areas. The conclusion of hardware performance from the measurement results heavily depends on the visitor's activity on the visiting device. In order to get consistent results, the user of the application has to run the measurement while ensuring all other activity on the device consume as less hardware resources as possible.

### **5.2.3. Comparison with Numion Stopwatch**

Numion Stopwatch is an online application which aims to measure the time spent by a web browser to load a web page. (Kessels, 2005) There is a small JavaScript script embedded in its front page waiting to run on a visitor's computer. A visitor to the front page of the application will find an input field in the middle of the screen for the user to enter the web address of the web page to be measured. After entering a web address and pressing the execution button, the web page will start being loaded into a sub frame of a web page which is a controller of the measurement process.

The controller page measures the time between the moments when a browser starts to load the URL and when the browser dispatches a signal to indicate that it has finished. The measurement therefore includes fetching and interpreting all HTML DOM elements, page frames, images, and dynamic language scripts. It does not include content that is handled by plugins. If the page is in the browser's cache, the measurement results will be based on the faster loading time.

The design of measurement process of Numion Stopwatch tool is quite similar with the implementation of the experimental part of the study. Both of the applications try to do the measurement of page loading time by allocating the measuring web page into a sub frame of a scripted web page which controls the operational progress.

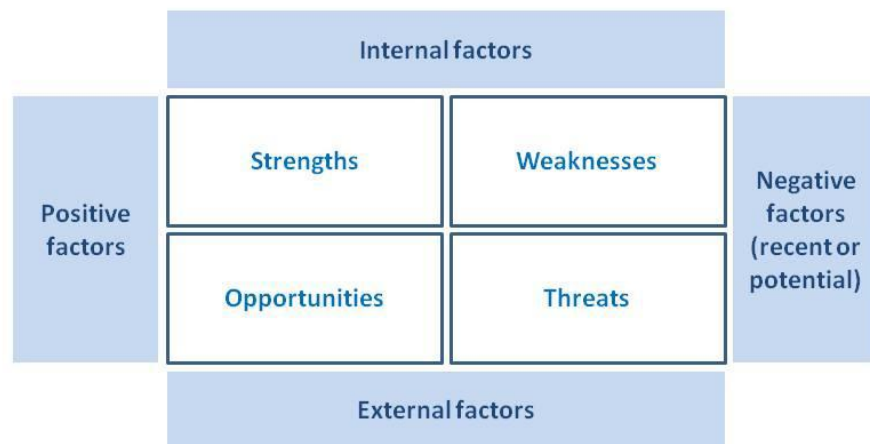
However the measurement tool of Numion Stopwatch can perform the measurement on only one web page for each time. The user has to input the web addresses one by one if there are a lot of web pages needs to be measured. Since Numion Stopwatch does not provide a hosting server to centralize web pages for measurement, the page loading speed is influenced heavily by the network connection quality. The usage of embedded frame to load an arbitrary web page will definitely meet troubles brought by incompatible web sites, which have been observed and handled carefully in the implemented application in the experimental part of the study. [Section 3.5.2]

## 6. Conclusion

The objectives of this study were defined in research questions and one of them was to find out consistent, efficient and reliable performance evaluation process for mobile browsers. The possibility to apply this measurement process to the productive environment for achievement on business was presented here in brief. At the end of this chapter, the challenges of the study are discussed and the research process will be evaluated. The possible future work is also discussed.

### 6.1. SWOT Analysis

Using SWOT to analysis the business outlook of a research project in development and its generated products is a popular practice in the market. SWOT Analysis is a strategic planning method used to evaluate the Strengths, Weaknesses, Opportunities, and Threats involved in a project or in a business venture. It involves specifying the objective of the business venture or project and identifying the internal and external factors that are favorable and unfavorable to achieving that objective. SWOT is an ideal way to measure business demand and identify the possibility of making profits from the market. By using SWOT, the business opportunity is easier to be understood. Here is a paragraph which describes the typical SWOT analysis results.



**Figure 6-1:** The SWOT model (CIPD, 2008)

Here the results of the experimental part of the study will be analyzed for the business prospects by using the SWOT analysis methods. The purpose is to find out the potentials to migrate the application structure to serve more users and to achieve commercial benefits if possible. The listed concerns are copied directly from the SWOT analysis template, (Chapman, 1995-2010) and the according analysis results are based on the implementation in the experimental part of the study.

## Strengths

Advantages of proposition:

The measurement environment is completely located in the cloud so that no investment is required.

Capabilities:

Concise code structure. Ease of maintenance. Wide scope of appliance for mobile devices.

Competitive advantages:

Free distribution of source code.  
Hosting on cloud computing platform.

Unique selling points:

App engine services within Google's infrastructure.  
Online access to source code for free.

Resources, assets, people:

Individual work-out. Software version control in use.  
Development tools for free.

Experience, knowledge, data:

Distributed Version Control System (DVCS) in use.  
Rich documentation available on the Internet.

Financial reserves, likely returns:

Online page advertising is possible. Possible to be adopted for enterprise use.

Marketing - reach, distribution, awareness:

Online open access to the implementation and source code. No budget on advertising yet.

Innovative aspects:

Performance evaluation with accuracy and efficiency.  
Modular design and code implementation.

Location and geographical:

The measurement environment is located in cloud and accessible worldwide.

Price, value, quality:

Free access. Use for free.  
Ease for maintenance and redeployment.

Accreditations, qualifications, certifications:

Online application proves successful.

Processes, systems, IT, communications:

No need for extra care. Everything is in the cloud.  
Ease for maintenance.

Cultural, attitudinal, behavioral:

Ease for localization.  
Ease for extending user amount.

Management cover, succession:

Individual work.  
Ease of knowledge transfer.

## Weaknesses

Disadvantages of proposition:

Specific on page loading speed measurement.  
Result presentation is too simple.

Gaps in capabilities:

Adding new feature requires re-do of the software engineering process.

Lack of competitive strength:

Measurement specifically on page loading speed.  
Accuracy is possible to be increased.

Reputation, presence and reach:

There are seldom notices on its existence so far.

Financials:

No budget for the implementation at all.  
No income source available in the near future.



Own known vulnerabilities:

The measurement results are presented awkward.  
Too many similar measurement tool in market.

Timescales, deadlines and pressures:

Single person product relies on a personal schedule.

Cash flow, start-up cash-drain:

No budget at all.

Continuity, supply chain robustness:

Depends on the interests from users and developer.

Effects on core activities, distraction:

Incompatible but popular web sites are big trouble  
to be measured.

Reliability of data, plan predictability:

As time goes by there will be better measurement  
tools available.

Accreditations, qualifications, certifications:

The security concern has not been included in the  
solution yet.

Morale, commitment, leadership:

Single person product.

Accreditations, etc:

Too many failure examples of Web 2.0 sites.

Processes and systems, etc:

Dependence on accessibility to Google servers.

Management cover, succession:

Knowledge was not shared in team.  
Too much knowledge to learn for a newbie.

## Opportunities

Market developments:

More functionality could be added.

Competitors' vulnerabilities:

Higher budgets.

More payment for human resources.

Industry or lifestyle trends:

Mobile users would like to try this for free.

Technology development and innovation:

Users enjoy trying new things online.

Global influences:

Accessible from anywhere in the world.

New markets, vertical, horizontal:

Device manufacturers might be interested in adopt the idea.

Geographical, export, import:

Accessible from anywhere in the world.

New unique selling points:

Google's cloud computing platform. Zero investment in the beginning.

Business and product development:

Sell idea to mobile device manufacturers.

Information and research:

Possible to add more functionality on App Engine.

Partnerships, agencies, distribution:

Combined with Google service.

Volumes, production, economies:

Free of charge for hosting and code upgrading.

## Threats

- Political effects: Google services not accessible in specific locations.
- Legislative effects: Content filter might be necessary.
- Environmental effects: None.
- IT developments: Experienced web developer is necessary for maintenance.
- Competitor intentions –various: There are already a lot of measurement tools available on the Internet.
- Market demand: Mobile device manufacturers might be interested in adopt this idea.
- New technologies, services, ideas: Maintenance work might be a lot to adopt to business environment.
- Vital contracts and partners: Google App Engine might be not free any more on some day.
- Sustaining internal capabilities: Another experienced web developer is needed.
- Obstacles faced: incompatible but popular web sites.
- Insurmountable weaknesses: lack of better presentation for measurement results.
- Loss of key staff: If there has no another experienced web developer coming.
- Sustainable financial backing: Google Adsense might not work.

## 6.2. Challenge of the study

During the preparation period of this research, it took a very long time for grasping the necessary concepts and knowledge for exerting development tools well enough. Since the online application development and cloud computing technologies evolve rapidly, a developer has to learn new concepts and coding skills in a short time which is relatively difficult for an amateur who can only do the researches in spare time.

The measurement of the page load performance of mobile browsers is a new topic for web browser performance evaluation. All mature evaluation solutions available in the market so far are designed specifically for desktop computing environment. As a result, although the page load performance of mobile browser is critical to end users, mobile device manufacturers do not invest enough on specific performance measurement tools.

In the implementation the referenced web pages are mirrored from the front pages of most popular web sites globally. However the category of these web pages might be different to what a mobile user would like to visit.

For different locations in the world people may visit quite different web sites daily. For this reason it would be better to define the categories of web pages based on the web surfing customs of people from different geography areas, and execute the measurement to simulate the page loading progress of their favorite web sites accordingly.

Another variety of web page content comes from the popularity on customized web sites for specific mobile devices. A number of popular web sites re-designed and implemented a second web interface based on their mobile users, removed some unimportant content from the revised version of the page to increase the page loading speed. This study did not take care of this situation since it is complicated to specify all customized versions of a web page. And there will be more such a kind of customization happening for new released mobile devices.

For those incompatible web pages which were failed to load in a sub frame embedded in a web page, most of them contain quite many media data and flash content to fill every pixel of a web page. Such a situation is quite common to see in some front pages from web sites hosting in China. It seems that the popularity of web sites did not increase the responsibility of their maintainers to follow good web site design principles.

### **6.3. Possible future work**

As the experimental part of the study concentrated on implementing a practical work model of the identified measurement process to evaluate the page load performance of mobile browser, some advanced features which should be available in such a measurement application are absent from the implemented work. This section listed some of these features in order to provide a reference for possible future development of the tool.

#### **Measurement results in graph**

The measurement results stored in database server module after each round of measurement are simply presented in form of tables so far, which do not give intuitive answers to the users without downloading and re-processing these data.

The future work should focus on providing an option for user to check the measurement results in a more intuitive way. For instance, the measurement results should be possible to be shown as the curves presented in the Result chapter, which indicated the variety among different mobile browsers clearly.

#### **More web pages as reference**

In the implemented application of the experimental part of the study, there are only 20 web pages available for reference. It is quite less than the amount of web pages referenced in a productive measurement environment. In order to enlarge the user group of this measurement tool, the web page server module of the application model should contain more web pages mirrored from actual web sites.

#### **Automatic detection of trouble pages**

During a measurement process of current implemented measurement application, the measurement progress will terminated with a zero value returned as the result if there is a web page incompatible to the HTML embedded frame. The whole measurement round has to be re-executed and the wrong result value has to be manually removed.

It should be possible for the measurement control point to monitor the error happened on the page loading progress and store necessary information of the problematic web page. The measurement process should not halt and the value of measurement results should be accessible together with the stored error information available to the user.

## Bibliography

Alan Grosskurth, M. W. (2006). *Architecture and evolution of the modern web browser*. Elsevier Science.

Ambler, S. W. (2010). *Mapping Objects to Relational Databases: O/R Mapping In Detail*. Retrieved April 30, 2010, from [www.agiledata.org](http://www.agiledata.org): Techniques for Successful Evolutionary / Agile Database Development: <http://www.agiledata.org/essays/mappingObjects.html>

Apple. (2011). *Apple*. Retrieved May 2, 2011, from <http://www.apple.com/>

Ari Jaaksi, N. (2002, July/August). Developing Mobile Browsers in a Product Line. *0740-7459/02 IEEE* (pp. 73-80). IEEE Software.

Bennett, J. (2006, July 2). *Django and Ajax*. Retrieved April 26, 2010, from [b-list.org](http://www.b-list.org): <http://www.b-list.org/weblog/2006/jul/02/django-and-ajax/>

BuiltWith. (2010). *JavaScript Usage Statistics*. Retrieved May 1, 2010, from [Web and Internet Technology Usage Statistics: http://trends.builtwith.com/javascript](http://trends.builtwith.com/javascript)

Chapman, A. (1995-2010). *SWOT Analysis*. Retrieved May 3, 2010, from [businessballs.com](http://www.businessballs.com): <http://www.businessballs.com/swotanalysisfreetemplate.htm>

CIPD. (2008, March). *SWOT analysis*. Retrieved May 3, 2010, from [cipd.co.uk](http://www.cipd.co.uk): <http://www.cipd.co.uk/subjects/corpstrtgy/general/swot-analysis.htm>

Django Software Foundation. (2005-2010). *Django | The Web framework for perfectionists with deadlines*. Retrieved April 18, 2010, from <http://www.djangoproject.com/>

DocForge. (2010, April 18). *Web application framework*. Retrieved April 26, 2010, from [DocForge: http://docforge.com/wiki/Web\\_application\\_framework](http://docforge.com/wiki/Web_application_framework)

Evans, C. C. (2004). *The Official YAML Web Site*. Retrieved April 18, 2010, from <http://www.yaml.org/>

Facebook. (2010). *Welcome to Facebook*. Retrieved May 3, 2010, from [Facebook: http://www.facebook.com/](http://www.facebook.com/)

Fielding, R. T., & Taylor, R. N. (2002). Principled Design of the Modern Web Architecture. In R. T. Fielding, & R. N. Taylor, *Principled Design of the Modern Web Architecture* (pp. 115-150). ACM Transactions on Internet Technology, Vol. 2, No. 2, May 2002.

Gardner, R. (2010, July 9). *Mobile Browser Usage on the Increase*. Retrieved May 2, 2011, from <http://netprofitstoday.com/blog/mobile-browser-usage-on-the-increase/>

Garrett, J. J. (2005, February 18). *Ajax: A New Approach to Web Applications*. Retrieved April 26, 2010, from adaptive path:  
<http://www.adaptivepath.com/ideas/essays/archives/000385.php>

Google. (2008). *Google App Engine*. Retrieved April 18, 2010, from Google Code:  
<http://code.google.com/appengine/>

Google. (2008). *The webapp Framework - Google App Engine*. Retrieved April 18, 2010, from Google Code: <http://code.google.com/appengine/docs/python/tools/webapp/>

Google. (2010). *Project Hosting on Google Code*. Retrieved May 3, 2010, from Google Code:  
<http://code.google.com/projecthosting/>

Google. (2010). *Quotas - Google App Engine*. Retrieved May 1, 2010, from Google Code:  
<http://code.google.com/appengine/docs/quotas.html>

Google. (2010). *Using Templates - Google App Engine*. Retrieved April 30, 2010, from Google Code:  
<http://code.google.com/appengine/docs/python/gettingstarted/templates.html>

Gürgens, F. (2011). *Test online your browser speed and CPU performance*. Retrieved May 25, 2011, from SPEED-BATTLE: <http://www.speed-battle.com>

Harvard Law. (2003, July 15). *RSS 2.0 Specification (RSS 2.0 at Harvard Law)*. Retrieved May 3, 2010, from Berkman Center For Internet & Society at Harvard University:  
<http://cyber.law.harvard.edu/rss/rss.html>

Hunt, A. (2010, 1 13). *Dont Repeat Yourself*. Retrieved April 18, 2010, from c2.com:  
<http://c2.com/cgi/wiki?DontRepeatYourself>

iPhone. (2011). *iPhone*. Retrieved May 2, 2011, from <http://www.apple.com/iphone/>

jQuery. (2006). *jQuery: The Write Less, Do More, JavaScript Library*. Retrieved April 18, 2010, from <http://jquery.com/>

jQuery. (2010). *Browser Compatibility*. Retrieved April 26, 2010, from jQuery JavaScript Library: [http://docs.jquery.com/Browser\\_Compatibility](http://docs.jquery.com/Browser_Compatibility)

JSON. (1999). *Introducing JSON*. Retrieved April 25, 2010, from <http://www.json.org/>

Kessels, J. (2005). *Numion Stopwatch*. Retrieved May 25, 2011, from <http://www.numion.com/Stopwatch/index.html>

Mazzetti, P., Nativi, S., & Bigagli, L. (2008). Integration of REST style and AJAX technologies to build Web applications; an example of framework for Location-Based-Services. *Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference on* (pp. 1 - 6). Univ. of Florence Prato, Florence: IEEE Conferences .

Mell, P., & Grance, T. (2009, October 7). *The NIST Definition of Cloud Computing*. Retrieved April 18, 2010, from NIST.gov - Computer Security Division - Computer Security Resource Center: <http://csrc.nist.gov/groups/SNS/cloud-computing/index.html>

Miller, J. (2009, February). *Design For Convention Over Configuration*. (Microsoft) Retrieved April 18, 2010, from MSDN Magazine: <http://msdn.microsoft.com/en-us/magazine/dd419655.aspx>

Netscape Communications. (2008). *The Netscape Archive*. Retrieved May 24, 2011, from <http://browser.netscape.com>

Prototype. (2006). *Prototype JavaScript Framework: ease development of dynamic web applications*. Retrieved April 25, 2010, from <http://www.prototypejs.org/>

Python Software Foundation. (1990-2010). *Python Programming Language -- Official Website*. Retrieved April 18, 2010, from <http://www.python.org/>

Rails. (2004-2010). *Ruby on Rails*. Retrieved April 18, 2010, from <http://rubyonrails.org/>

Selfa, D., Carrillo, M., & Del Rocio Boone, M. (2006). A Database and Web Application Based on MVC Architecture. *Electronics, Communications and Computers, 2006. CONIELECOMP 2006. 16th International Conference on* (pp. 48 - 48). Benemerita Universidad Autónoma de Puebla, Mexico: IEEE Conferences.

Shan, T. C., & Hua, W. W. (2006). Taxonomy of Java Web Application Frameworks. *e-Business Engineering, 2006. ICEBE '06. IEEE International Conference on* (pp. 378 - 385). Shanghai: IEEE Conferences.



SpringSource. (2004-2010). *SpringSource.org*. Retrieved April 18, 2010, from <http://www.springsource.org/>

SunSpider. (2011). *SunSpider JavaScript Benchmark*. Retrieved May 25, 2011, from <http://www.webkit.org/perf/sunspider/sunspider.html>

Thomas, D. (2003, March 10). *Orthogonality and the DRY Principle*. (i. b. Venners, Editor) Retrieved April 18, 2010, from artima developer: <http://www.artima.com/intv/dry.html>

Twitter. (2010). *Twitter - Discover what's happening right now, anywhere in the world*. Retrieved May 3, 2010, from twitter.com: <http://twitter.com/>

W3C. (2010). *Cascading Style Sheets*. Retrieved May 3, 2010, from W3C.org: <http://www.w3.org/Style/CSS/>

W3C. (2010). *World Wide Web Consortium*. Retrieved May 3, 2010, from <http://www.w3.org/>

Ye, P. (2010). Research on Mobile Browser's Model and Evaluation. *978-1-4244-6359-6/10 IEEE* (pp. 712-715). 2010 IEEE.

## Appendices

### Appendix A: Source Code of the Experimental Work

The source code of the experimental part of the study has been uploaded and managed by the online version control service provided freely by Google. (Google, 2010) The code resources and all pictures used on the user interface are located in the URL below. The whole research work is released under GNU GPL v2.

<http://code.google.com/p/meixi/source/browse/#hg/ptest>

Here in the appendix all source code files are not listed. Only the important source files are listed for providing reference to the research evidence. Other unlisted source code files can be accessed from the URL above.

#### File: app.yaml

```
application: pageloadspeed
version: 1
runtime: python
api_version: 1

handlers:
- url: /sites
  static_dir: local_sites

- url: /static
  static_dir: static

- url: /favicon.ico
  static_files: static/favicon.ico
  upload: static/favicon.ico

- url: /*
  script: view.py
```

## File: model.py

```
import yaml
from google.appengine.ext import db

PRESET = yaml.load(file('preset.yaml', 'r'))

class Case(db.Model):
    name = db.StringProperty()
    core = db.StringProperty()
    device = db.StringProperty()
    connection = db.StringProperty()
    scope = db.StringProperty()
    created = db.DateTimeProperty(auto_now_add=True)
    ip = db.StringProperty()

    def getOne(self):
        case = {}
        case['key'] = str(self.key())
        case['name'] = self.name
        case['result'] = Record().speedSum(self)
        case['core'] = self.core
        case['device'] = self.device.split()[-1]
        case['connection'] = self.connection.split()[-1]
        case['scope'] = self.scope.split()[-1]
        case['created'] = str(self.created).rsplit(':', 1)[0]
        ips = self.ip.split('.')
        ips[1] = ips[2] = '-'
        case['ip'] = '.'.join(ips)
        return case

    def getList(self):
        list = []
        query = self.all().order('-created')
        for x in query:
            list.append(x.getOne())
        return list

    def remove(self):
        Record().removeBy(self)
        self.delete()

    def removeAll(self):
        query = self.all()
        for x in query:
            x.remove()

class Record(db.Model):
    case = db.ReferenceProperty(Case)
    pagerank = db.IntegerProperty()
    speed = db.IntegerProperty()

    def getList(self, case):
        list = []
        query = self.all().filter('case = ', case).order('pagerank')
        for x in query:
            record = {}
            record['rank'] = x.pagerank
            record['site'] = PRESET['sites'][x.pagerank - 1]
            record['speed'] = x.speed
            list.append(record)
        return list

    def speedSum(self, case):
        sum = 0
        query = self.all().filter('case = ', case)
        for x in query:
            sum += x.speed
        return sum

    def removeBy(self, case):
        query = self.all().filter('case = ', case)
        for x in query:
            x.delete()
```

## File: preset.yaml

```
devices:
- Nokia N900
- Nokia N950
- Nokia N97 Mini
- Apple iTouch
- PC

connections:
- WiFi 802.11g
- WiFi 802.11n
- 3G
- Ethernet

scopes:
- Small - 5
- Midium - 10
- Large - 20

sites:
- google
- facebook
- youtube
- yahoo
- blogger #5
- baidu
- wikipedia
# live
- twitter
# qq
# msn
- yahoo_jp
# sina
# google_in
# taobao
- amazon #10
- linkedin
# google_hk
- wordpress
# google_de
- bing
# google_uk
- yandex
- ebay #15
# google_jp
- microsoft
# google_fr
# netease
# google_br
# google_user
- flickr
- fc2
- mail_ru
# google_it
- craigslist #20
```

## File: view.py

```
import os
os.environ['DJANGO_SETTINGS_MODULE'] = 'settings'
from google.appengine.dist import use_library
use_library('django', '1.2')
from google.appengine.ext import webapp
from google.appengine.ext.webapp.util import run_wsgi_app
from google.appengine.ext.webapp import template
from django.utils import simplejson
from model import *

class MainView(webapp.RequestHandler):
    def get(self):
        values = {
            'devices': PRESET['devices'],
            'connections': PRESET['connections'],
            'scopes': PRESET['scopes'],
            'cases': Case().getList(),
        }
        path = os.path.join(os.path.dirname(__file__), 'templates/main.html')
        self.response.out.write(template.render(path, values))

class CaseClean(webapp.RequestHandler):
    def get(self):
        Case().removeAll()
        self.redirect('/')

class CaseView(webapp.RequestHandler):
    def get(self):
        act, key = self.request.path.strip('/').split('/')
        try:
            case = db.get(db.Key(key))
        except db.Error:
            self.redirect('/')
            return

        if act == 'read':
            values = {
                'case': case.getOne(),
                'records': Record().getList(case),
            }
            path = os.path.join(os.path.dirname(__file__), 'templates/case.html')
            self.response.out.write(template.render(path, values))
        elif act == 'run':
            values = {
                'case': case.getOne(),
            }
            path = os.path.join(os.path.dirname(__file__), 'templates/run.html')
            self.response.out.write(template.render(path, values))
        elif act == 'delete':
            case.remove()
            self.redirect('/')

    def post(self):
        name = self.request.get('name')
        core = self.request.get('core')
        device = self.request.get('device')
        connection = self.request.get('connection')
        scope = self.request.get('scope')
        ip = self.request.remote_addr

        case = Case(
            name=name,
            core=core,
            device=device,
            connection=connection,
            scope=scope,
            ip=ip)
        case.put()

        self.response.headers['Content-Type'] = 'application/json'
        self.response.out.write(simplejson.dumps(str(case.key())))

class CreateRecord(webapp.RequestHandler):
    def post(self):
        key = self.request.get('key')
```

```

        results = self.request.get('results').split(',')
    try:
        case = db.get(db.Key(key))
    except db.Error:
        self.redirect('/')
        return

    for i, x in enumerate(results):
        record = Record(case=case, pagerank=i+1, speed=int(x))
        record.put()

class SiteList(webapp.RequestHandler):
    def get(self):
        self.response.headers['Content-Type'] = 'application/json'
        self.response.out.write(simplejson.dumps(PRESET['sites']))

class Redirect(webapp.RequestHandler):
    def get(self):
        self.redirect('/')

application = webapp.WSGIApplication(
    [
        ('/', MainView),
        ('/caseclean', CaseClean),
        ('/create', CaseView),
        ('/read/.+', CaseView),
        ('/run/.+', CaseView),
        ('/delete/.+', CaseView),
        ('/record', CreateRecord),
        ('/sitelist', SiteList),
        ('.*', Redirect)],
    debug=False)

def main():
    run_wsgi_app(application)

if __name__ == "__main__":
    main()

```

## File: static/main.js

```
$(function() {
  getCore();
  $('input#name').val('New Run');
  $('#submit').click( function() {
    var name = $('input#name').val();
    var core = $('input#core').val();
    var device = $('select#device option:selected').val();
    var connection = $('select#connection option:selected').val();
    var scope = $('select#scope option:selected').val();
    $.post('/create', { 'name': name,
                       'core': core,
                       'device': device,
                       'connection': connection,
                       'scope': scope},
          function(key){ window.location='/run/' + key; }
    );
  });
});

function getCore() {
  $.each($.browser, function(i, val) {
    if(val === true) {
      $('input#core').val(i + ' ' + $.browser.version);
      return false;
    }
  });
}
```

## File: static/run.js

```
$(function() {
  $('#div#hint').hide();
  $('#iframe#subframe').load( function() { stopWatch(); });
  key = document.location.href.split('/').pop();
  rank = 0;
  scope = parseInt( $('#td#info_scope').text() );
  sites = [];
  results = [];
  $.getJSON('/sitelist', function(list) {
    sites = list;
    if(scope > sites.length) scope = sites.length;
    $('#div#hint').show();
    startWatch();
  });
});

function startWatch() {
  $('#span#timer').text((rank + 1) + '/' + scope);
  start_time = new Date();
  $('#iframe#subframe').attr('src', '/sites/' + sites[rank] + '.html');
}

function stopWatch() {
  stop_time = new Date();
  var speed = stop_time - start_time;
  $('#iframe#subframe').attr('src', '');
  results.push(speed);
  if(++rank < scope) {
    startWatch();
  } else {
    $("#iframe#subframe").unbind('load');
    $.post('/record', {'key': key, 'results': ' ' + results}, function() {
      $("#div#hint").append(' .. Done. ');
      window.location = "/";
    });
  }
}
```



## File: templates/case.html

```
<!DOCTYPE html>

<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
{% include "title_style.html" %}
</head>

<body>
{% include "info.html" %}
<table>
<tr>
<th>Rank</th>
<th>Website</th>
<th>Page load speed</th>
</tr>
{% for record in records %}
<tr>
<td>{{ record.rank }}</td>
<td>{{ record.site }}</td>
<td>{{ record.speed }}</td>
</tr>
{% endfor %}
</table>

</body>
</html>
```

## File: templates/info.html

```
<h5>PLPE - Page Load Performance Evaluation(Experimental)</h5>
{% if case %}
<table>
<tr>
<th>Name</th>
<th>Core</th>
<th>Result</th>
<th>Device</th>
<th>Connection</th>
<th>Scope</th>
<th>IP</th>
</tr>
<tr>
<td>{{ case.name }}</td>
<td>{{ case.core }}</td>
{% if case.result %}
<td>{{ case.result }}</td>
{% else %}
<td> ... </td>
{% endif %}
<td>{{ case.device }}</td>
<td>{{ case.connection }}</td>
<td id="info_scope">{{ case.scope }}</td>
<td>{{ case.ip }}</td>
</tr>
</table>
<p />
{% endif %}
```

## File: templates/main.html

```
<!DOCTYPE html>

<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
{% include "title_style.html" %}
<script type="text/javascript" src="/static/jquery.min.js"></script>
<script type="text/javascript" src="/static/main.js"></script>
</head>

<body>
{% include "info.html" %}
<table>
<tr>
<td>Name:</td>
<td><input id="name" type="text" /></td>
</tr>

<tr>
<td>Browser Core:</td>
<td><input id="core" type="text" readonly="readonly" disabled="disabled" /></td>
</tr>

<tr>
<td>Device:</td>
<td>
<select id="device">
{% for device in devices %}
<option>{{ device }}</option>
{% endfor %}
</select>
</td>
</tr>

<tr>
<td>Connection:</td>
<td>
<select id="connection">
{% for connection in connections %}
<option>{{ connection }}</option>
{% endfor %}
</select>
</td>
</tr>

<tr>
<td>Scope:</td>
<td>
<select id="scope">
{% for scope in scopes %}
<option>{{ scope }}</option>
{% endfor %}
</select>
</td>
</tr>

</table>

<div><input type="submit" value="RUN" id="submit"/></div>
<p /><p />

{% if cases %}
<table id="case_list">
<tr>
<th><a href="/caseclean">C</a></th>
<th>Name</th>
<th>Result</th>
<th>Device</th>
<th>Connection</th>
<th>Scope</th>
<th>Time</th>
<th>IP</th>
</tr>
{% for case in cases %}
<tr>
```

```
<td><a href="/delete/{{ case.key }}">x</a></td>
<td>{{ case.name }}</td>
<td><a target="_blank" href="/read/{{ case.key }}">{{ case.result }}</a></td>
<td>{{ case.device }}</td>
<td>{{ case.connection }}</td>
<td>{{ case.scope }}</td>
<td>{{ case.created }}</td>
<td>{{ case.ip }}</td>
</tr>
{% endfor %}
</table>
<p />
{% else %}
<p>The list of cases is empty.</p>
{% endif %}

</body>
</html>
```

## File: templates/run.html

```
<!DOCTYPE html>

<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=utf-8" />
<meta http-equiv="expires" content="0" />
<meta http-equiv="Pragma" content="no-cache" />
<META HTTP-EQUIV="CACHE-CONTROL" CONTENT="NO-CACHE" />
{% include "title_style.html" %}
<script type="text/javascript" src="/static/jquery.min.js"></script>
<script type="text/javascript" src="/static/run.js"></script>
</head>

<body>
{% include "info.html" %}
<div id="hint">Page load running ... <span id="timer" /></div>
<iframe id="subframe" width="80%" height="80%" />

</body>
</html>
```

## File: templates/title\_style.html

```
<title>PLPE - Page Load Performance Evaluation(Experimental)</title>
<style type="text/css">
body {font-size: 1.2em;}
input, select {font-size: 1.1em;}
select {width: 10.5em;}
th {background-color: #66ff99;}
th, td {padding: 4px 16px 2px 16px;}
td {background-color: #ddd;
    text-align: right;}
tr:nth-child(odd) td {background-color: #ccc;}
</style>
```