Evaluating, and scaling web programming languages: A comparison between PHP, JavaScript

Mahmood F. Abdullah
Assistant Lecturer

mahmoodfawzi@yahoo.com

Abstract

This work concerns with comparing web server resources. The most known web programming languages (PHP-JavaScript) were presented. An age calculator implemented as a web application. Many readings were taken as response time, CPU load and other readings are examined and compared. PHP was found to give best results in some cases and JavaScript is more suitable for other cases.

Keywords php, javascript, performance

تقييم، و تحجيم لغات البرمجة الشبكية:

مقارنة بين PHP, javaScript

محمود فوزي محمود

مدرس مساعد

مستخلص

يتعلق البحث بمقارنة موارد خوادم الشبكات. تم استخدام لغات برمجة الشبكات الاكثر شهرة (PHP, JavaScript). وتم استخدم الة حساب العمر كتطبيق في هذه الخوادم. تم استخلاص عدة قراءات منها زمن الاستجابة، الحمل على وحدة المعالجة المركزية و قراءات اخرى كثيرة درست وتم المقارنة بينها. وقد اعطت لغة (PHP) افضل النتائج في بعض الحالات. اما لغة (JavaScript) فقد كانت ملائمة لحالات اخرى.

Introduction

Just as there is a diversity of programming languages available and suitable for conventional programming tasks, there is also a diversity of languages available and suitable for Web programming. There is no reason to believe that any language will completely monopolize the Web programming scene, although the varying availability and suitability of the current offerings is likely to favor some over others. PHP and JavaScript are used every day for sync or a-sync interaction. PHP is a computer scripting language, originally designed for producing dynamic web pages [1]. It is mainly used in server-side scripting. It is a widely-used general-purpose scripting language that is especially suited for web development and can be embedded into HTML. It generally runs on a web server, taking PHP code as its input and creating web pages as output. It can be deployed on most web servers and on almost every operating system and platform free of charge [2]. PHP is installed on more than 20 million websites and 1 million servers, although the number of websites with PHP installed has declined since August 2005[3]. JavaScript is a scripting language most often used for client-side web development. It was the originating dialect of the ECMAScript standard. It is a dynamic, weakly typed, prototype-based language with first-class functions. JavaScript was influenced by many languages and was designed to have a similar look to Java, but be easier for non-programmers to work with. The language is best known for its use in websites (as client-side JavaScript), but is also used to enable scripting access to objects embedded in other applications (for example Microsoft Gadgets in Windows Vista Sidebar). Despite the name, JavaScript is essentially unrelated to the Java programming language, though both have the common C syntax, and JavaScript copies many Java names and naming conventions [4].

A real experience to check the power of web programming for both languages is done by testing both languages with same application the NeoLoad. The NeoLoad is a professional load testing software provides all the features needed to carry out a load tests and analyze the results, all from a unique and integrated interface. By simulating large number of users an application simultaneously and to check response time under the predicted load, with a predefined scenario for simultaneous number of users[5]. A program which was written to be tested was the loan calculator which has the ability to check the total number of days from the date (dd/mm/yy) that the user has provide till date of test and obtain the age of the user at the end of the test. The programming mechanism for

such application is done by getting the date (day-month-year) provided by the user, calculating number of days since the user's date, and finally providing the result. For the PHP code which is located at the server side (Apache-server) see Fig.(1),the procedure will be as follows:

1. The client requests the first page which contains the input form.
2. The server will receive a page request from the client, and replay the requested page to the client.
3. The user at the client side will put the necessary input and post the request again to the server.
4. Then the server will execute the PHP program producing the result as an HTML page with the number of days, then replaying the HTML page as a result to the client.

The JavaScript program is located inside the HTML page which is loaded to the client machine since the HTML page is requested from the web server (Apache-server). The user at the client machine will provide the date (day-month-year) which is passed to the JavaScript program to be processed within the client machine not the server as in the PHP, then providing the result as a HTML page, or in the same original date post page, Fig.(2).

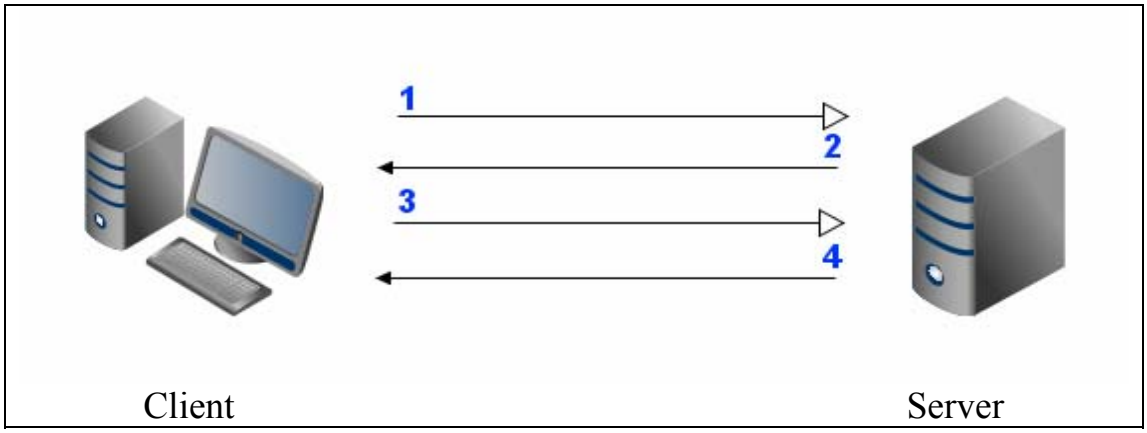

Fig.(1). PHP, client server, requests & responses.

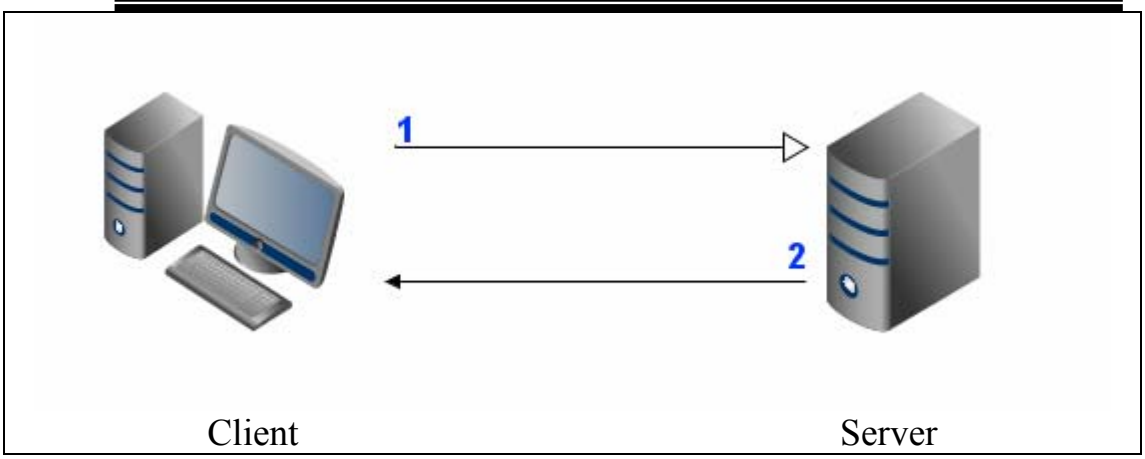

Fig.(2).JavaScript, client server, request & response.

Implementing the scenario

The basic idea of the scenario for the experiments of both the PHP and Java-Script web pages is almost the same; the first scenario for the Java-Script was planed as the user should request the webpage at the web browser. Then the HTML document with the embedded Java-script code will be loaded to the client's web browser. Then the user provides the required information (Year, month, day). After proceeding with the (calculate) function the web browser read all information provided at the fields to examine the total number of days starting from the date provided.

The second scenario with the PHP web site starts with loading the web page when the user request it, and after all required information were provided. The provided information will be transferred back to the web server to be calculated by PHP at the server, then a HTML result page will be generated by PHP and loaded back to the client web browser.

Then both scenarios are executed separately, average pages which is the total number of pages requested, average hits which is the Number of successful hits, total pages requested, total hits, average page response time, average page request response time, total throughput(the average rate of successful message delivery over a communication channel) [6], and total users launched all are recorded. Both scenarios were planned to run with load of 10 clients, within duration of 10 minutes. Table (1) (Statistics Summary) provides a brief summary for the specified readings, where column A represents reading for the PHP scenario, and column B represents readings for the JavaScript scenario:

Table (1) Statistics Summary

%	B	A	
+99.3-%	3.3	485.7	**Average hits/s**
+99.3-%	1985	291891	**Total hits**
+57.1-%	0.003s	0.007s	**Average Request response time**
+0%	0	0	**Total hit errors**
+76.9-%	0.003s	0.013s	**Average Page response time**
+99.4-%	0.01 Mb/s	1.55 Mb/s	**Average throughput**
+0%	0	0	**Total action errors**

As observed in table one the following results were recorded for both scenarios:

1. Average hits pages/s for PHP scenario recorded 291891 hits/s while JavaScript scenario recorded 1985 hits/s, with (0%) total hit error for both scenarios, as shown in Fig.(3).
2. Average Request response time for the PHP scenario took about the double time with (0.007s) compared to the JavaScript scenario time (0.003s) indicating (57.1%) increase in Request response time compared to the JavaScript scenario, as shown in Fig.(4).
3. Average Page response time in the PHP scenario took about (0.013s) while the JavaScript scenario took about (0.03s), as shown in Fig.(5).
4. Average throughput in JavaScript scenario recorded (0.01Mb/s), while the PHP scenario took about (1.55Mb/s) with total increase of (98.6%), as show in Fig.(6) .
5. No errors were recorded in both scenarios, as shown in Fig.(7).

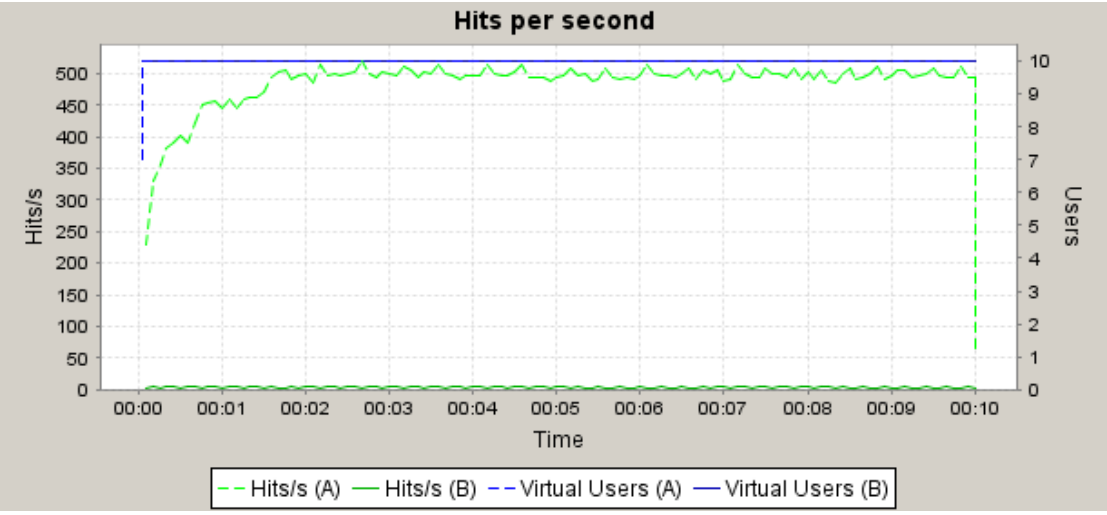

Fig.(3).Hits/s

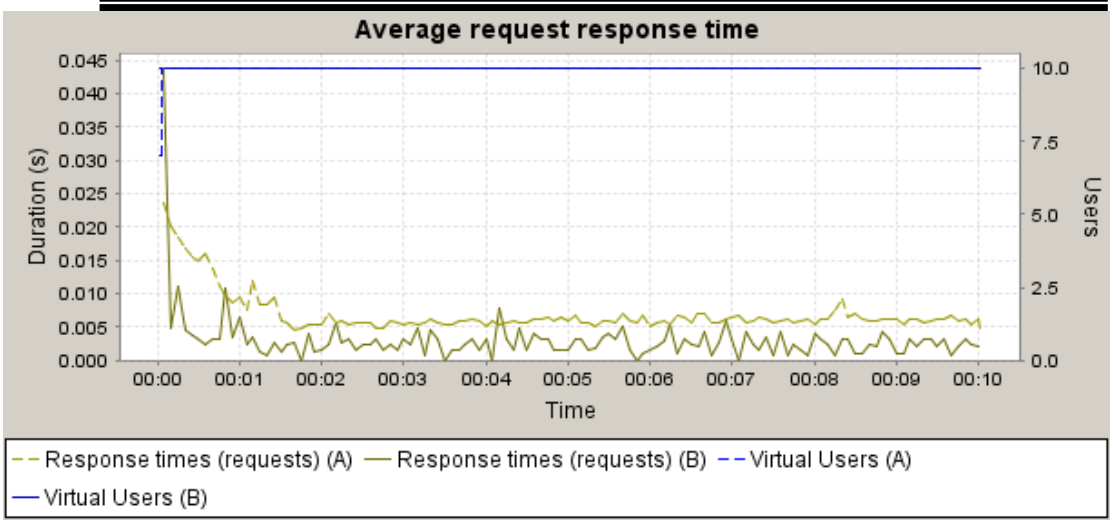

Fig.(4).Average request response time.

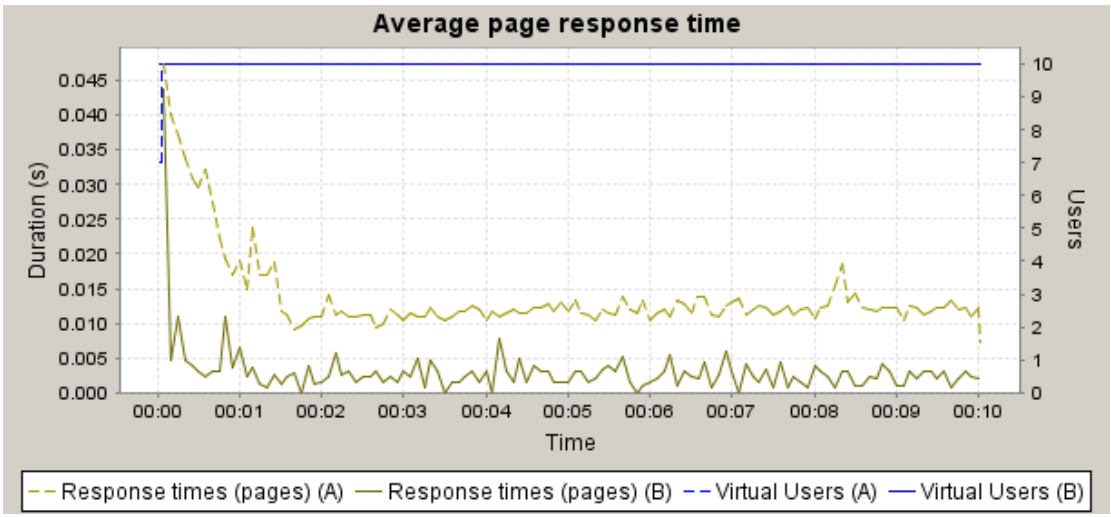

Fig.(5).Average Page response time.

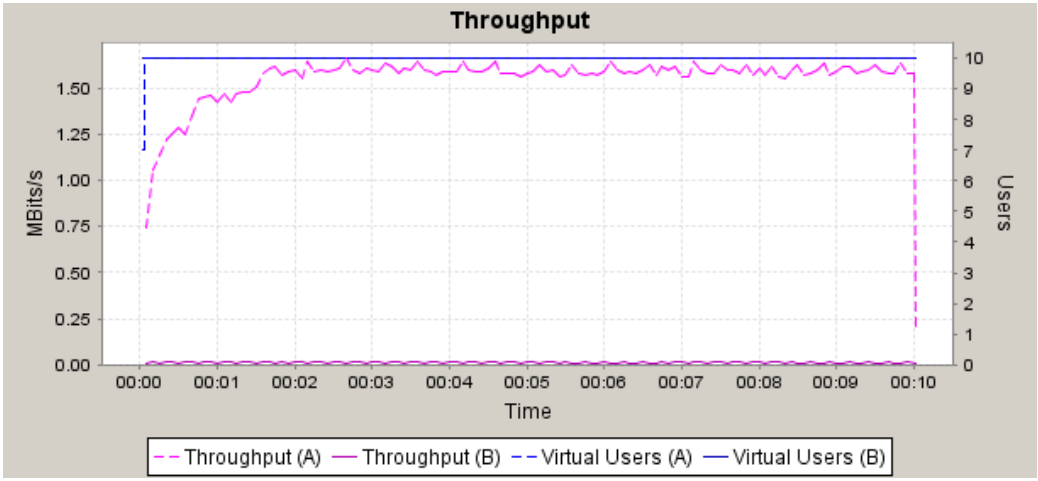

Fig.(6).Throughput.

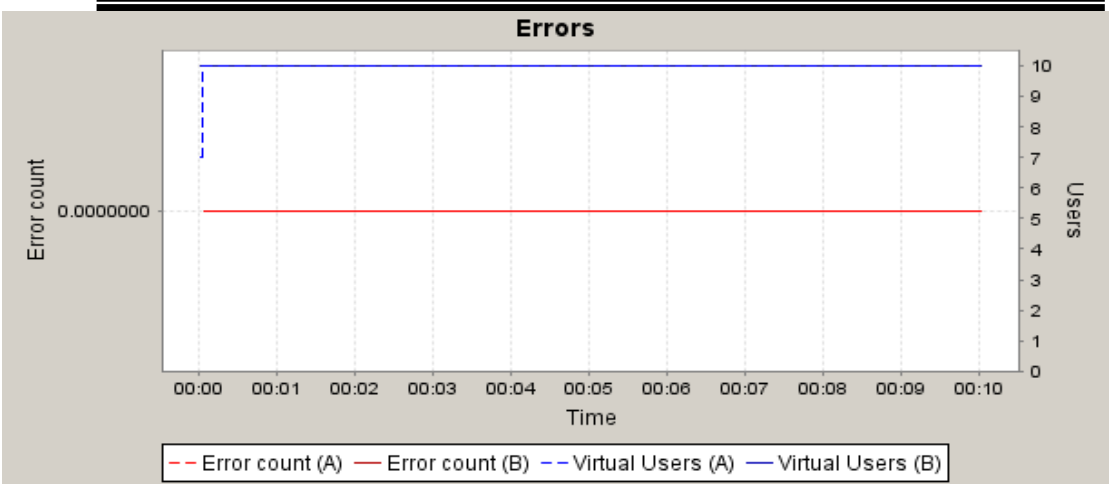

Fig.(7).Total errors.

with initializing the NeoLoad to calculate readings for the client resources, including system Memory ,process time, CPU IO Data Bytes, CPU IO Read operation/sec, the following readings were taken as shown in table two(Performance counters):

Table2.Performace counters

Max %	Max B	Max A	Avg %	Avg B	Avg A	Min %	Min B	Min A
Server/windows/Memory/Available M Bytes								
+25.4%	84.00	67.00	+50%	78.00	52.00	+58.3%	76.00	48.00
Server /windows/Processor/% Processor Time								
+85.9-%	14.00	99.00	+90.8-%	9.00	98.00	+97.9-%	2.00	96.00
Server /windows/Process/IO Data Bytes/sec								
+88-%	2049019.00	17010115.00	+96.9-%	491126.00	16017937.00	+98.5-%	138997.00	8991454.00
Server /windows/Process/IO Read Operations/sec								
+98.4-%	4228.00	264260.00	+98.6-%	3404.00	248836.00	+98.5-%	2043.00	139390.00

Conclusions:

1. Most of the readings proved that the PHP web application consumes many hardware resources for the server, like CPU, and memory.
2. As PHP language runs on the server, there was an increase in the network activity between the client and the server, and this is not recommended for low bandwidth networks.
3. Huge and specialized applications like Databases can not be handled by JavaScript while PHP language can deal with these applications.
4. As PHP works at the server side, all requests can be handled smoothly despite of the large number of requests simultaneously, as the web server should be a dedicated machine with special specifications.
5. In any case the security at the client side is weaker than in server side.

References

[1] PHP Server-Side Scripting Language. Indiana University. 2007-04-04.

[2] Adam Trachtenberg, David Sklar,"PHP Cookbook: Solutions and examples for PHP Programmers", Paperback - Aug 1, 2006.

[3]http://afruj.wordpress.com/2008/05/19/history-of-web-dev-programming-languages/

[4] http://en.wikipedia.org/wiki/JavaScript.

[5]http://www.neotys.com/load-testing/how-it-works.html.

[6] Blahut, Richard E. "Algebraic Codes for Data Transmission", Cambridge University Press, 2004, ISBN 0521553741