Is Server-Side Programming Killing Your Web Server?

Costas Vassilakis & Giorgos Lepouras
University of Athens
Department of Informatics
TYPA Buildings
Panepistimiopolis
Athens 157 71
Greece
Phone: (301) 7275220
Fax: (301) 7275214
Email: {costas, G.Lepouras}@di.uoa.gr

Introduction

The paper addresses issues related to client/server technologies and specifically the effectiveness of server-side programming techniques. The motive for this study was the need to create a lightweight and dynamic navigational aid for use in a web site. Towards this goal a number of possible solutions were considered and for two of them an experiment was run to determine the best suited for our case. The rest of the paper outlines our findings.

Motivation

Our objective was to provide an easy way of navigating through the information of a web site. The requirements set were to design and implement a navigational aid that would be flexible, extensible, light, multilingual and generic enough to be supported by most web clients. With the old navigation system users had to go down three or four levels of information in order to reach the desired data. If they did not know exactly what they were looking for, they could easily be lost in the "info-maze". To this end, a hierarchy tree was necessary to enable the user to determine his/her location and navigate more easily.
Most users are accustomed to using a menu, therefore a menu style navigational aid was thought to be the most appropriate. The pages could be categorised according to their content and grouped together accordingly. A possible categorisation could have the form of:

- University
- Department
    - General
        - History
        - Sectors
        - Staff
    - Education
        - Courses
        - Timetable
    - Research
        - Groups
        - Publications
        - Technical Reports

- Theses

Possible solutions

A number of alternative solutions were examined before reaching a final decision. If all items of the hierarchy tree were always visible, then the overwhelming number of categories would tire and confuse the user. So, it was necessary to construct the means for dynamically expanding or collapsing hierarchy categories.
Java is portable, flexible and can be used to implement interactive menu structures that are executed on the client's computer. However, the files produced by the Java compiler are rather large, increasing download time. The initial download time may be compensated for, if the user is going to work at the site for a long period of time, but occasional visitors are discouraged from visiting the site. Additionally, Java is not supported by older and text-based browsers, and supporting non-Latin fonts requires extensible preparation, such as locale-specific resource bundles, native to ASCII transformations etc.
A second alternative, currently employed by numerous sites is JavaScript. Since JavaScript requires the use of images to model active areas, there is no problem displaying international characters. Moreover, JavaScript code is usually simpler (at least for low complexity tasks) than the equivalent Java. On the other hand the use of images penalises download time. Furthermore, apart from not being supported by older and text-based browsers, programmers have to take into account various differences in the JavaScript programming interface offered by web browsers.
Server-Side Programming is another technique for implementing dynamically created html pages. This feature can be exploited to develop interactive navigational aids. The major advantage of this approach is that it can produce standard HTML, ensuring browser independence. International character sets can be catered for by including the appropriate META (*charset=*) tag, while older browsers can be configured to support these sets by specifying the corresponding font. The disadvantage of this solution is the extra processing load imposed on the Web Server.
The following table summarises the advantages and disadvantages of the examined solutions. Other solutions such as ActiveX and Dynamic HTML were rejected because they can not be used with all server or client platforms.

	Light	Portable	Dynamic	Intl. Characters
Java	-	+	++	+
JavaScript	-	-	+	+
Server-Side Programming	+	++	+	+

Chosen solution

The initially chosen solution was to implement a hierarchical menu that would be created dynamically on the server each time the server received a request. This would provide a light, interactive, menu-type navigational aid for users, which could be easily configured to reflect the site's needs. A small program was developed, which reads through a menu resource file and creates the appropriate html output. Since, the number of possible output files was predetermined, a question was risen as to whether this solution was as efficient as creating beforehand all possible output pages. This procedure could be automated by means of a "compilation" program that would read through the menu resource file and generate all possible expanded and collapsed menu pages. To test this case an experiment was set up to evaluate the appropriateness of each approach under different workloads.

Experiment

To this end, two different configurations were set up, both using the Apache Web Server version 1.3.2. In the first configuration a Sun Sparc4 with 128 MB of RAM was used and clients connected to it via a medium-speed network. On the second configuration an IBM compatible with a 200MHz MMX Intel Processor and 64 MB of RAM running Linux was used and clients connected to it via a high-speed network. In both configurations two measurement sets were obtained for various concurrent request rates, ranging from one to ninety six requests. The first set involved dynamically generated HTML pages, while the second concerned pre-generated HTML pages. For dynamically generated HTML pages the response time (i.e. the time between the request submission and the server's reply) and the program execution time were measured. In this case, *program execution time* denotes only the time taken by the server-side program to generate the HTML page, which does not include any overheads such as process creation, inter-process communication etc. For pre-generated HTML pages only the response time was measured.

Results

The following figures illustrate the obtained measurements. The diagrams clearly show that the response time for pre-generated pages is significantly less than the corresponding time for dynamically created pages. Although this result was expected, the first issue worth noting is that the difference is not due as much to the program execution time as to the overheads imposed to the server by the server-side approach. This conclusion is backed up by the fact that while the order of magnitude for program execution time is milliseconds (see figure 3), the corresponding difference in response time (see figures 1, 2) is measured in seconds. The imposed overheads include:

- Process creation, management and synchronisation, since the server must spawn a new process for each request.
- Housekeeping activities, such as environment setup before program execution and memory allocation/de-allocation.
- If a CGI approach is employed, inter-process communication costs are incurred for communicating the query string to the program; furthermore, the output of the CGI programs is never cached, thus request rates and server load increase even further. If a server-side include option is selected the disk file containing the relevant directives must be read and analysed.

Finally, the operating system itself may introduce overheads, due to paging and swapping phenomena, process switching, etc.

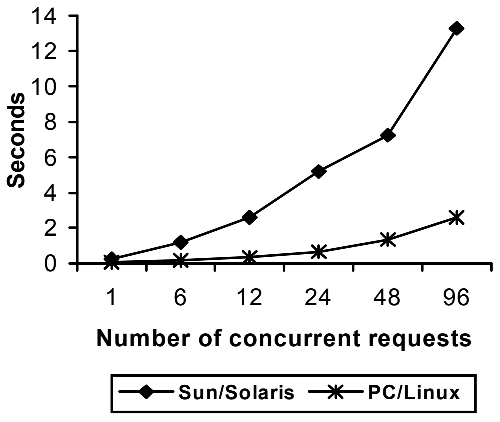

Figure 1 – Response time for dynamic pages

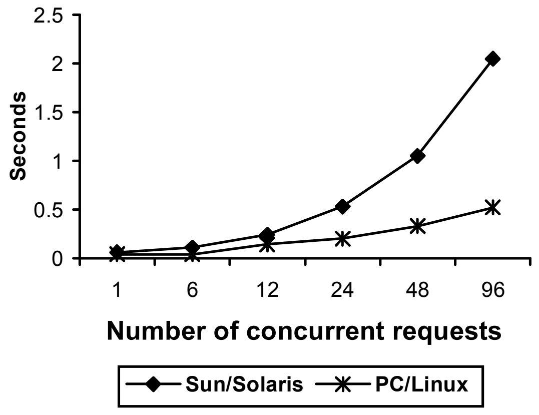

Figure 2 - Response time for pre-generated pages

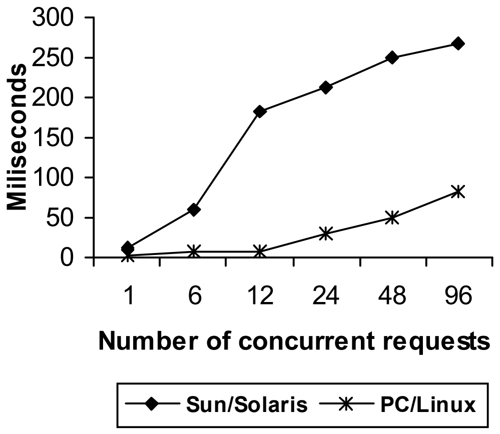

Figure 3 – Program execution time for dynamic pages

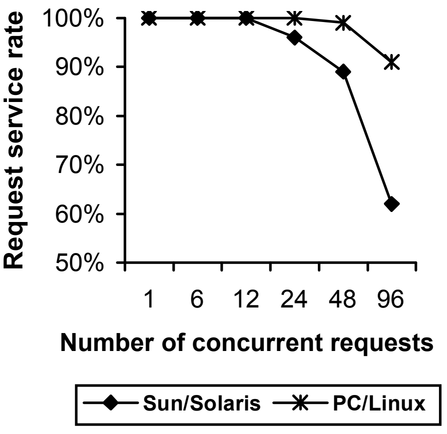

Figure 4 – Request service rate for dynamic pages

Another issue worth noting is that when dynamic pages are used, the server *begins to reject requests* when a certain load limit is reached (see figure 4). This load limit is dependent on the machine architecture and the operating system, but it is clear that the server saturation limit is reached with significantly less load when dynamic pages are used (in our experiments, the saturation threshold was not reached with pre-generated pages, even when the number of concurrent requests was set to 200).

Conclusions

Our experiments have shown that server-side programming should be exercised with caution on web servers, especially when performance is a major consideration. In order to select between dynamically generated and pre-computed HTML pages, several parameters have to be taken into account. Clearly using pre-computed pages is inapplicable when the number of possible output pages is too large or their contents cannot be pre-determined. Insufficiency of disk resources and high update frequency of the underlying data also advice against such a solution. On the other hand, in cases that the anticipated number of concurrent requests is high, or the computer hosting the web service is slow or overloaded, or the time needed to compute each output page is excessive, using pre-generated pages is favoured. Furthermore, using a "compilation" program to produce the pre-generated pages enhances the maintainability of this scheme, since the web administrator only needs to modify the underlying data and rerun the compilation program, instead of determining and modifying all affected pages.