Where the Speed Matters… Zero-Response-Time Search Engine for Small Collections

Ruwan Gamage

School of Information Management, Wuhan University, China
and
Library, University of Moratuwa, Sri Lanka
ruwan@lib.mrt.ac.lk

Abstract. Users with slow internet connections experience slow retrieval of results in web catalogues. JavaScript search engines can be used to enable client side search, reducing the load on the server, and increasing the response time. However, it is not a popular method until now, because of various reasons including limitation of number of data objects and lengthier response time for the first search. Here the author suggests negotiating the issue of response delay with the user. This would enable high speed basic search in small catalogues, usually with less than 300 data objects. Larger catalogues can be divided into smaller ones. Special or rare collections, multimedia artifacts and subject (web) directories are prospective candidates for this type of search systems. A prototype catalogue of 'Sri Lankan Web Sites' was tested in www.srilankasupersearch.com. Users' behavior and response to the system is yet to be studied.

Keywords: JavaScript; search engines; response time; negotiation; OPAC; models.

1 Introduction

Web based digital libraries and web catalogues offer search tools for users to mine information from databases. Most of these databases offer server side handling of search queries. In this type of systems, users experience a delay in retrieval of results. This delay is termed as response time, latency or lag time. Technically, response time refers to the amount of time it takes for input from a keyboard to reach the application and a response returned.

Length of 'response times' depends on various factors. Response time in a network is usually proportional to the number of users currently using the network, the location of the network components, and the complexity of the network.

Higher the response time, it is more embarrassing for the user. Previous research suggests that user productivity is dramatically reduced when response time is significantly longer. Sterbenz [1] states that further productivity gains are realized

E.A. Fox et al. (Eds.): ICADL 2005, LNCS 3815, pp. 224 – 231, 2005.

when the response time decreases to the range of 100 ms. According to him, human factors studies have also indicated that consistent response time is better for users than response with a significant variance, since users alter their behavior based on response time at a relatively slow rate.

Nielsen [2] confirms that 0.1 second (100 ms) threshold is suitable while 1.0 second limit is acceptable. Within this limit users' flow of thought will be uninterrupted. Ten seconds is the limit the user can focus his attention.

These observations were used to create a model for enabling high speed client side search for a very small data set.

1.1 Client Side Processing

In contrast to client-server systems, client side search strategies mainly depend on the performance of the client computer and browser. Therefore it is quite fast to handle a search request, rather than transforming the load on to the server.

JavaScripts use this strategy efficiently. A JavaScript search engine can be used to imitate OPACs with comparatively smaller collections. Data objects for search can be arranged in an array within the JavaScript. The JavaScript then creates cookies on client machine. However the time needed to create cookies depend on the number of elements in the array. If the number of elements in the array is more, the JavaScript becomes heavy, taking a lot of time to create cookies on the client machine. A suggestion for negotiating this time lag with user is described here.

1.2 Other Attempts to Increase Response Time or Negotiating with the User

Most of the other attempts described here are meant for large databases. Though these can not be compared with this model, it will give an idea on the quest for reducing response time.

Chan and Ueda [3] focused on using cached objects with enough information to connect back to the server to request more information. However, such solutions require resources to be held open on the server, waiting for client responses. Long [4] introduces a query slicing technique which would display data in sets, not as a whole.

One other approach for searches is to build web agents. Web agents will search for information on behalf of the user, according to his preferences. Such preferences are stored in a user profile database. It has a learning function and can learn the users' likes and dislikes when the user searches the web with keyword searching thus reducing the response time for the next search [5].

Sterbenz [1] advices the programmer to display the reason for the delay, which the user can read while he waits for the result/expected page. He further proposes on running the more complex operation in a new window. That leaves the user the original page for working with, until he gets the search result.

1.3 Overview of the Search Engine

While JavaScript search engines are easy to write, and there are many predefined ones openly available on the web, the author used JSE Search, an open source Java Script [6].

JSE is platform independent and doesn't require .NET, ASP, CGI or any other technologies on host. JSE circumvents HMTL's inability to pass a value from one page to another by using a session or non-persistent cookie. The cookie expires when the user's session ends.

JSE consists of two scripts (see Appendix). The first writes a cookie containing the search words and then loads the results page. The second script does all the work; reading the cookie, defining the matches and generating the search results. Results set is a single page numbered list with links to detailed pages if available. For users, searching is similar to Google. A preceding minus character excludes a word, while phrases are supported within double-quotes.

The cookie is stored in browser's memory. Therefore subsequent searches experience virtually zero response time compared to the first search.

1.4 Data Array

Data is held as a JavaScript array, within the search.js JavaScript file. Within the array, each line consists of a data object and its description, URL, etc. If the number of data objects is high, size of file becomes larger. This will cause a delay in creating cookies on browser. That results in a delay in the response time of the first search of the session.

2 Search Model

As explained above, using a JavaScript for searching means there is a delay between the first search request and displaying of the first result. Though the delay in the first search is high, the response times of subsequent searches become virtually zero. Therefore there should be a method to negotiate with the user until the first search is carried out. The strategy used here for this is first allowing the user to do a configuration before doing the actual search. This configuration is actually a pseudo search (See fig.1).

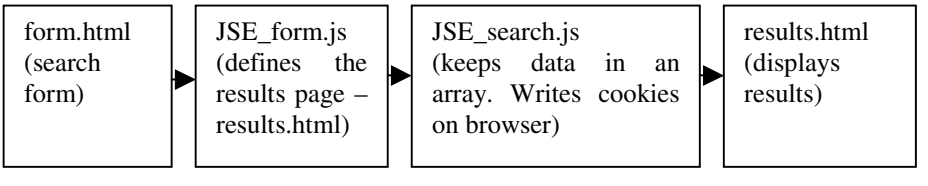

Fig. 1. Existing model for *JSE Search*

In order to achieve the pseudo search, one layer of action (2 files) was added to the existing model (See fig. 2, 3 & 4).

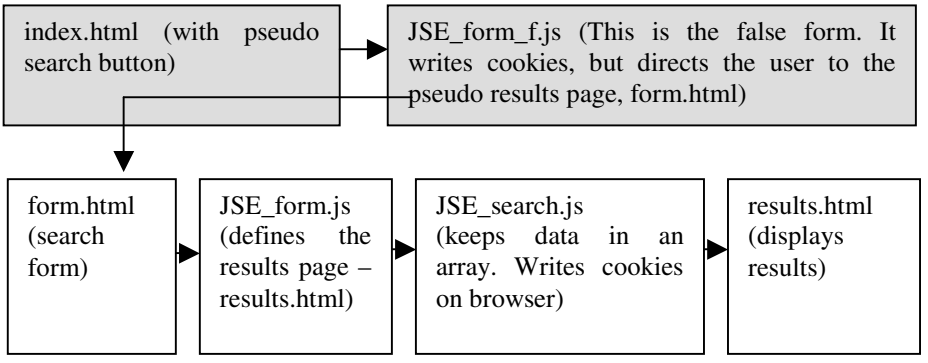

Fig. 2. Proposed model for negotiating with user to stay until *cookies* are created on browser

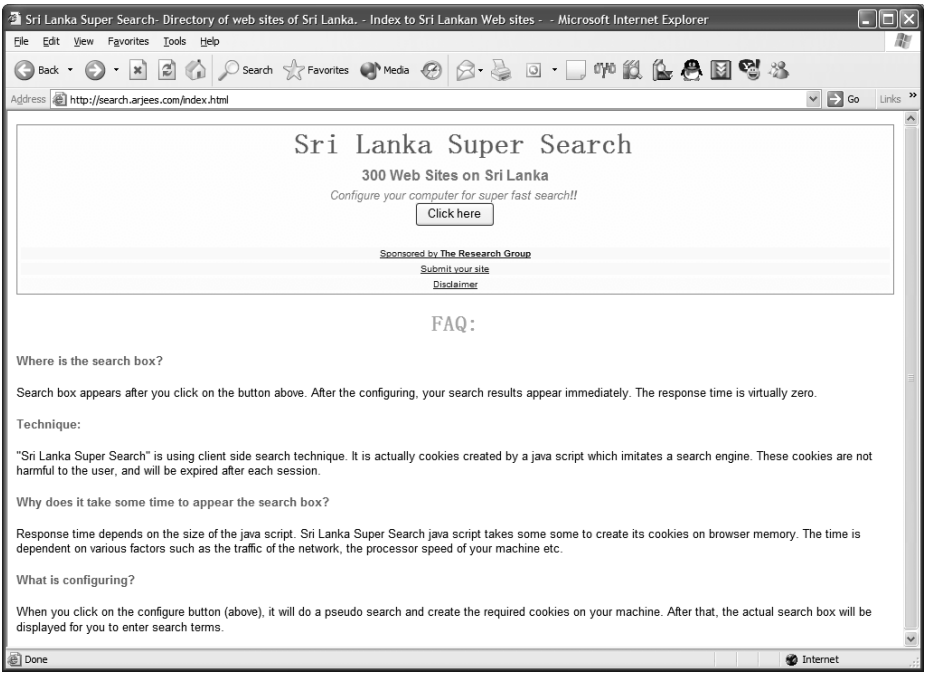

Fig. 3. Home page of *www.srilankasupersearch.com* requesting the user to configure the system before search

When the pseudo search finishes its function, the following page appears where the user can carry out his own search.

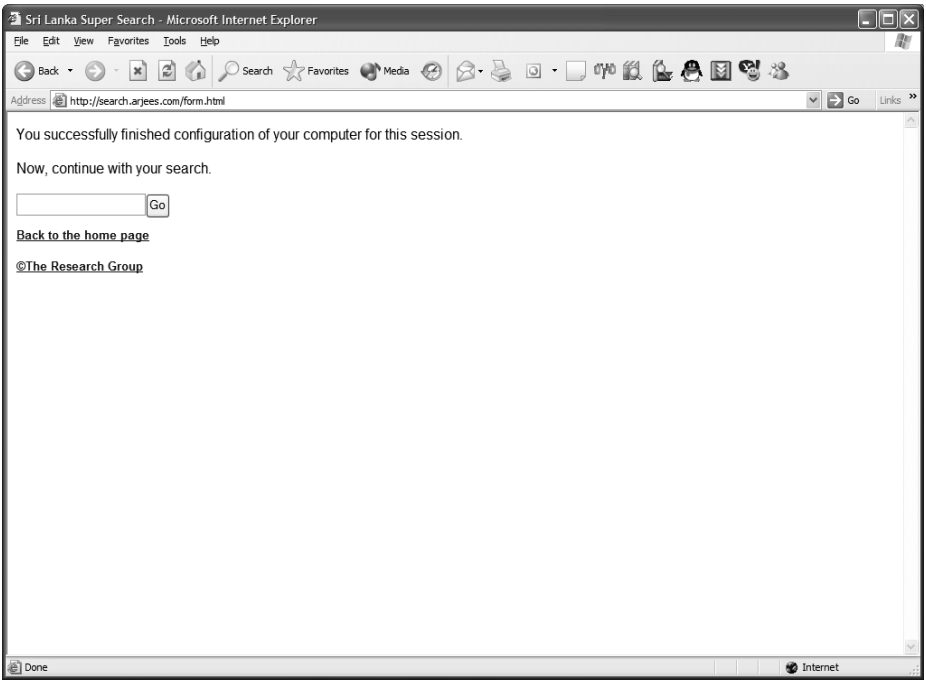

Fig. 4. Actual search page

3 Results

Response times were measured for different sized search.js files. All measures were taken using an Intel Celeron (900 MHz) machine within a half an hour time period to avoid differences in web traffic during different times of day. The results are as follows.

Table 1. Change of response time with size of the JavaScript

Number of data elements in the JavaScript array	Size of JavaScript file (kb)	Initial Response time - server side search (seconds)	Subsequent response times - client side search
1025	350	177	*
500	165	140	*
400	129	103	*
300	105	40	*

* Virtually zero

As this is a negotiated task, an empirical threshold of 40 seconds; 4 times than the limitation of keeping attention stated by Neilson [2], was considered to be acceptable.

Therefore only 300 data objects were considered for the web search. In other words, the service is for searching within a sample of 300 items.

4 Limitations, Improvements, and Uses

4.1 Limitations

Security threats, limitation of size of the file, requirement of expertise and labor to write the JavaScript are shortcomings in the present model. Also, this model can be used only with browsers which support cookies. Also, if the user has disabled script activities, the interface will simply ignore the request of the user.

Also, it should be noted that the above response times can not be standardized because these may change in other situations dependent on various factors including processor speed, internet connection, etc. which are beyond the scope of this article.

There are many disadvantages of the display method and display screen. Inability to prioritize results is a major draw back. However, already there are JavaScript search engines with many advanced features as the one given by Bradenbaugh [7]. He also claims that the JavaScript search engine he presents has carried nearly 10000 data elements without much problem.

OPAC data should be ones which can be made publicly available, because outsiders can easily download the whole JavaScript file with data, from the server. However, linked full text or other source files can be housed in the sever using ASP or other technologies, giving more security.

4.2 Improvements

Whether the user accepts this model or not is yet to be decided based on user behavior studies. If the user agrees to wait for a long time, the number of data objects can be further increased. This will also be based on the importance of the set of data objects for the user.

Writing Javascripts appears troublesome. However this can be automatically written using open source software like WinISIS, one which is familiar to library professionals. Header and footer be the same, while it is only the data objects *(array)* which is added to the script. It is a matter of creating a good *'print format'*.

4.3 Advantages and General Use

One of the benefits against a server side search engine is, this can be tested even on a stand alone computer as it is. Debugging is easy. The same JavaScripts, along with search form and results page can be housed in a real server. Therefore even school and small public libraries can house their OPACs on web. Because higher technologies such as .Net, ASP or CGI are not needed, even a free web server can be used to host the OPAC.

Because the server is made free after the first search of each session (configuration), the server can accommodate more requests from first time users. The server-side traffic is lower than server side search engines.

If the database is so large, it can be separated into well defined categories. Search can be enabled within these categories.

The categories themselves can be identified as data objects. Therefore, an optional primary search can be made available to identify which categories have results related to the user's information need.

5 Conclusion

Using JavaScript search engines for imitating OPACs for very small collections can improve the customer satisfaction by giving speed access to search results. Demanding the user to search in several search spheres is apparently a backward option, but it is worthwhile compared with virtually zero response time in subsequent searches. This would enable catalogues already on web to improve their search options. Also, this model will be helpful to immediately host web OPACs for those who have not yet done so.

This would enable high speed basic search in a very small catalogue, usually with less than 300 data objects. Rare books, dissertations, multimedia artifacts and other special collections are prospective candidates for this type of search systems.

One should not always assume that using a JavaScript search engine means some amount of delay. If the number of data objects is very low, the system may retrieve even the first results set, very quickly.

References

1. Sterbenz, James P. G.: High-Speed Networking: A Systematic Approach to High-Bandwidth Low-Latency Communication. John Wiley & Sons, Incorporated, New York (2001) 441-442.
2. Nielsen, J.: The need for speed at http://www.useit.com/alertbox/9703a.html. 1997. Retrieved June 2005.
3. Chan E. and Ueda K.: Efficient Query Result Retrieval Over the Web. In Proceedings International Conference on Parallel and Distributed Systems. IEEE Computer Society (2000).
4. Long, B. A: Design Pattern for Efficient Retrieval of Large Data Sets from Remote Data Sources in on the Move to Meaningful Internet Systems, LNCS 2519 (2002) 650–660.
5. Quah, Jon T. S., Chen, Y. M., Leow, Winnie C. H.: in Chapter XVIII Networking E-Learning Hosts Using Mobile Agents Leow, Intelligent Agents for Data Mining and Information Retrieval. Idea Group Inc., Hershey (2004). 263-293.
6. JSE Documentation. Downloaded from http://www.JavaScriptkit.com/script/script2/jse/jse10a.zip on 30.05.2005.
7. Bradenbaugh, Jerry. JavaScript Cookbook. 1st Edition October 1999 (est.) O'Reilly.

Appendix: Form.js and Search.js (part) JavaScript Files

form.js

```
// ---------- script properties ----------
var results_location = "results.html";
// ---------- end of script properties ----------
function search_form(jse_Form) {
     if (jse_Form.d.value.length > 0) {
            document.cookie = "d=" + escape(jse_Form.d.value);
            window.location = results_location;
     }
}
```

search.js

```
// ---------- script properties ----------
var include_num = 1;
var bold = 0;
// ---------- sites ----------

var s = new Array();

s[0] =
"Autosrilanka.com^http://www.autosrilanka.com/^<blockquote><b>ht
tp://www.autosrilanka.com/</b>     >>>     Free     advertisements
(vehicles)</blockquote>^Autosrilanka       Advertising       free
advertisements    promotion    vehicles    auto    cars    vans
automobilespublicity promotions marketing sales";
s[1] =
"EeZee2.com^http://www.eezee2.com/^<blockquote><b>http://www.eez
ee2.com/</b>            >>>            Free            Classified
Advertisements.</blockquote>^EeZee2.com     Advertisingpublicity
promotions marketing sales";

// ---------- sites continue within this array----------

// ---------- end of script properties and sites ----------

var cookies = document.cookie;
var p = cookies.indexOf("d=");

// ---------- script continues ----------
--------
--------
--------
end.
```