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].