Issues in High Performance Full-Text Searching of Digital Documents

Cyrus Shaoul (cyrus@garage.co.jp)
Ted Crossman (tedo@garage.co.jp)

Digital Garage Inc.
http://www.garage.co.jp/

Abstract: Digital documents have become one of the most common means of communication for the human race in the latter half of the 20th century. As the number of digital documents available over data networks, such as the Internet, has increased, so has the difficulty in finding documents on these networks. Internet Search Engines have tackled the problem of allowing people to search the entire set of documents that make up the World Wide Web. The WWW is a set of documents that grows at a rate that is hard to measure, but may be over 100% per month. The content of the WWW is administered in a completely decentralized manner, and documents can change content, location or disappear completely at any time, usually with high frequency. We will try to review in this paper the issues that face a high performance search engine in this anarchistic environment.

Issues in High Performance Full-Text Searching of Digital Documents

The Internet poses new challenges to high performance full-text searching of digital documents. In terms of scope, currency, speed, size of document sets, and document formats, the requirements put upon search engines are enormous. These requirements have gone beyond the scope of "how fast can a document be found in an index". The speed of lookup in the index is still very important, but other parts of the problem have become equally important. In this paper we will try to concisely analyze what areas other than index design have become problem issues that face the designers of high performance search engines. The following areas appear to the authors to be the most relevant: Document Discovery, Index Currency, Index Content, and Index Management. For the purposes of this paper we will only consider the searching of webs of hypertext under the HTTP protocol.

Document Discovery

Document discovery is process where a search engine must seek out documents to add to its index. To be specific, in a web of hypertextual data, a search engine must perform link extraction and link filtering to efficiently find all documents that are required for a collection of documents. We will call the part of the search engine that discovers and retrieves data from a network the "spider".

Link extraction is the process by which a page of hypertext is processed, and the full list of links that are contained within that page are extracted. Care must be taken to correctly process relative and absolute links. There are many ways to use this list, of which two are: breadth first and depth first traversal of the web. Spiders are usually programmed to use one of these two approaches, or mixed approaches. The spider discovers new links and it must make decisions about whether the link should be included in the index. This brings us to the main issue in spidering: spider control. High performance spidering may not be possible without efficient link filtering.

Theoretically, using regular expressions and other means, the activities of the spider might be able to be controlled with fine precision. Some controls that could be used in link filtering in spiders are: wildcard URL patterns, number of directories in the URL, and the number of hops from first URL. A further type of filtering could be to allow only the links in a hypertext document into the list of future links to be visited, excluding the text in the document. The opposite type of filter could also be useful (allowing the text only, no links.) with a combination of these types of filters, a very detailed, complex rule-set could be developed, for example:

```
Start the Spider at the Root URL: http://www.osaka.ac.jp/index.html
```

```
Allow all URLs of the wildcard pattern: http://www.*.ac.jp/*
Disallow all URLs of the wildcard pattern: http://www.tokyo.ac.jp/*
All URLs must be 6 or less hops away from the Root URL
```

This type of theoretical filter set would collect all pages from web servers that have "www" as the first part of their hostname, and are in the ac.jp academic domain. It would exclude pages from the server www.tokyo.ac.jp, and will only collect pages to a depth of 6. The implementation of this type of filtering may not be particularly difficult, but if the number of rules grows over 50, and the spider is gathering 10,000 documents per hour, the implementation of high-performance filtering software becomes a significant issue.

With the size of the document collection growing at that rate, it is quite possible to assemble indexes of hundreds of thousands of documents in a short time. As stated before, with the reality of decentralized management of these documents, the next challenge of the search engine to maintain a current version of each document in the index.

Index Currency

To keep a large index current, the search engine must use very efficient methods of automatically tracking the volatility of documents in the index. The simplest, not very efficient method could be to check the state (modified/not modified) of every document in the index at fixed intervals. As the size of the index increases, the time taken to do such a check may grow at a linear rate. As the time it takes to do a complete check of the currency of the index increases, the overall currency of the index decreases due to number of documents that change during check period. For higher performance, and better currency, another method for tracking changes in the documents may be needed. This is the core problem in the issue of index currency.

There will be many different approaches to this problem, and the most interesting ones will be the ones that satisfy the following conditions: automatic detection of per-document frequency of change and automatic spidering of new or changed content based on frequency of change. Furthermore, the configurability of the search engine is also an issue. A high performance search engine needs to be able to be configured so that it can track changes in sets of documents that change at many different rates (twice a day, once every 11 days, and even once every year), with hundreds of documents changing every day. These performance issues are all key in automatically maintaining a current index. As we shall see in the next section, it is also important what data is indexed, and how it is indexed, so that high performance, complex queries are possible.

Index Content

The core of the index is terms and their locations. But in a high performance search

engine, other information can be added to the index to increase the efficiency and usability of the index. In this section we will enumerate non-textual types of data and meta-data that can by put in an index, and explain the rational behind their inclusion.

1) Document checksum
2) Date of modification
3) Document MIME type
4) Document site location [site=www.osaka.ac.jp]
5) Document link information [doc contains links to http://xx.xx.xx]
6) URL Information [doc URL is www.osaka.ac.jp/a/b/c.html]
7) META data information [doc contains data "author:Mark Twain"]

The need for this data to be included in the index is clear as we look at the needs of users and of the software itself. For users of the search engine, the ability to search both the full text of a document at the same time as searching the document's meta data. For example, the pseudo-query "find all documents that are from the site "this.test.co.jp" that are of MIME type application/acrobat that contain the phrase 'the size of the moon'". But the ability to search on the checksum and other types of data may also used by the spider itself, when it is looking in the index for duplicate documents. This synergism between the spider/indexer and the index itself could be one source of improved performance in search engines.

Index Management

The efficiency in which a search engine allows the management of its indexes is an increasingly important factor in the overall performance of the search engine. Index management consists of the at the minimum, the following tasks:

1) Adding Entries
2) Deleting Entries
3) Merging indexes

One issue in the performance of index management is the ability to do certain activities "live", that is to say without having to shut down or disconnect users from the search engine. A high performance search engine should be able to add and delete documents from an index in real-time without any apparent effect in the usability of the search engine. The same need exists for the process of merging indexes. As the scale of index merging increases, the ability of the search engine to merge million document indexes while still servicing search requests will become a required feature of a modern search engine.

Conclusion

The performance of index management in terms of speed and "liveness" of management operations is the final issue in search engine performance presented here, but not the final issue by any means, as there are many issues that we could not cover in this document. Still the summary we have presented here covers the basic issues that face creators of high performance full-text search engines today. The approaches that different designers will take towards optimization and invention will determine what technologies will become predominant in the near future.

References:

Infoseek Software (http://software.infoseek.com/)
Digital Garage/Ultraseek Server(http://software.infoseek.co.jp/)
Search Engine Watch (http://www.SearchEngineWatch.com/)