ISSN XXXX XXXX © 2017 IJESC

Research Article Volume 7 Issue No.6

Design of Compression and Indexing Techniques for Documents of Domain Specific Cluster

Shivani[1], Varun Kaushik[2]
M.Tech Student[1], Assistant Professor[2]
Department of CS/IT
MIET, Meerut, India

Abstract:
The World Wide Web is a huge source of hyperlinked information. Web is growing every moment in context of web documents. Search engines use web crawlers to collect web documents from web for storage and indexing. In the existing system, it has been observed that the repository resources are limited. So it has become an enormous challenge to manage the local repository (storage module of search engine) in a way to handle the web documents efficiently that leads to less access time of web documents and proper utilization of available resources. This paper proposes architecture of search engine with the clustered repository, organized in a better manner to make task easy for user to retrieve the web pages in reasonable amount of time. The thesis work focuses on cluster balancing issue by partitioning the repository into domain specific memory blocks called clusters. This technique forms the base for the division of repository into multiple blocks. The partitioning approach deals efficiently with the problem of limited size of storage resources and data searching complexity in repository. The main component of proposed architecture is coordinator module which not only indexes the documents but also decide the memory cluster for a web document but also uses compression technique to compress the size of document and increase the storage capacity of repository.

1. INTRODUCTION

Engineering a search engine [39] is a challenging task. Search engines index tens to hundreds of millions of web pages involving a comparable number of distinct terms. They answer tens of millions of queries every day. There are differences in the ways various search engines work, but they all perform three basic tasks:
1. They search the Internet or select pieces of the Internet based on important words.
2. They keep an index of the words they find, and where they find them.
3. They allow users to look for words or combinations of words found in that index.
A search engine finds information for its database by accepting listings sent in by authors who want exposure, or by getting the information from their web crawlers spiders or robots programs that roam the Internet storing links to and information about each page they visit. A web crawler is a program that downloads and stores Web pages, often for a Web search engine. Roughly, a crawler starts off by placing an initial set of URLs, *S*0, in a queue, where all URLs to be retrieved are kept and prioritized. From this queue, the crawler gets a URL (in some order), downloads the page, extracts any URLs in the downloaded page, and puts the new URLs in the queue. This process is repeated until the crawler decides to stop. Collected pages are later used for other applications, such as a Web search engine or a Web cache.
The most important measure for a search engine is the search performance, quality of the results and ability to crawl, and index the web efficiently. The primary goal is to provide high quality search results over a rapidly growing World Wide Web. Some of the efficient and recommended search engines are Google, Yahoo and Teoma, which share some common features and are standardized to some extent.
The critical look at the available literature reveals that the following points should be considered to develop a more efficient system.
(a) *Scalability* – Web is growing every instant and this growth of web challenges the repository to adopt large number of new web pages at every instant with its limited size.
(b) *Large updates* - Web changes every moment. Therefore, the repository needs to maintain high rate of modification. These changes occur in the form of web pages that is new version of web pages are added to web so the space occupied by old version must be reclaimed.
(c) *Obsolete pages* – At one side pages are added to web and at the other side pages are deleted from website but repository is not notified. So the must be mechanism to detect such obsolete pages and make space utilization for new updates.
(d) *Searching* – As repository is local collection web pages and searching within the local collection must lead to retrieve the web pages in the reasonable amount of time.

2. ARCHITECTURE

The proposed work is an architecture as shown in figure 4.2 of crawler with enhanced repository technique which deals efficiently with the problem of limited size of storage resources and data searching complexity in repository. This technique forms the base for the division of repository into multiple blocks. The memory is partitioned into numbers of fixed size blocks and these blocks are called as clusters as represented in figure 4.1 and the partition of memory into blocks is done on the foundation of different domains like .com, .edu, .org etc. that belongs to web. As the memory block creation is based on

specific domain so these clusters are also called as domain specific clusters. Each domain has its own cluster and the particular cluster contains the documents that only belong to its specific domain type. Partitioning the repository into number of clusters and balancing these clusters, two different modules i.e. coordinator module and ranking module are used.

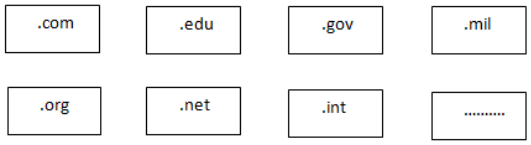

Figure.1. Domain specific clusters

Figure 2.2 depicts the architecture of the clustered web base system in terms of main functional modules and their interaction. There are mainly eight modules in architecture –

- The downloader
- The crawler
- The coordinator
- The ranker
- Indexer
- The query engine
- The search interface

The associations among these modules represent exchange of information as indicated by directions. The downloader module directly downloads the web pages from web and forwards it to crawler. The crawler module is responsible for retrieving pages from the web and handing these web pages to the storage management module that is coordinator module in proposed architecture. The crawler periodically goes out to the web to retrieve fresh web pages and latest copies of already existing pages in the repository. The coordinator module performs various critical functions that include Identifying document domain and format, assignment of page to decided cluster, extract URL, analyze URL, forward URL and stores the document in compressed form. Huffman compression algorithm is applied on the document to reduce their size. The proposed work focuses on the Coordinator module. The coordinator indexes different documents to indexer that help in searching process. Indexer module contains metadata of stored documents. The ranker module increases the rank of document basis of popularity to maintain the freshness on of repository and also help coordinator module to maintain the size of clusters.

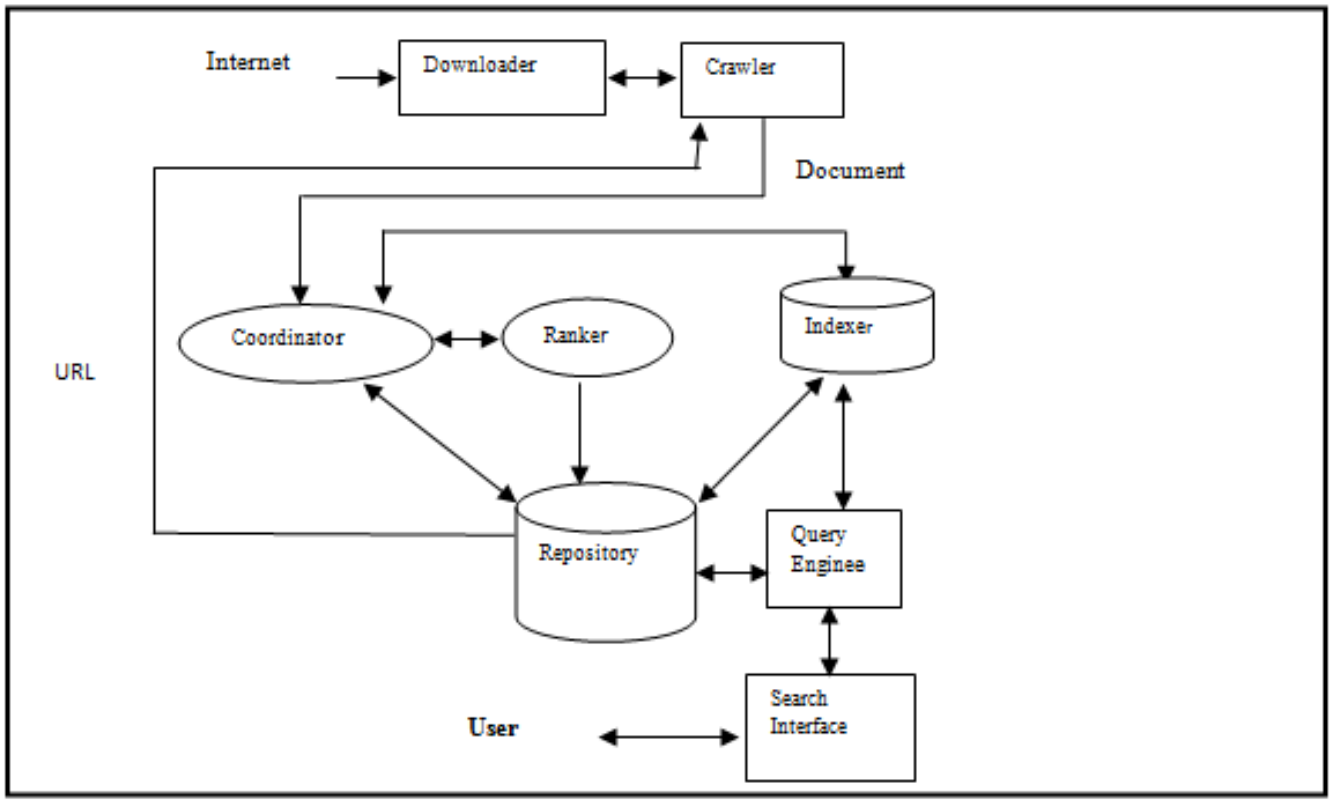

Figure.2. General Architecture of clustered Web base Search engine

The query engine module is responsible for receiving and filling search requests from users and the search module is that module to which user interact to request web page from web.

Description of the various modules

Functioning of Coordinator Module

The most important component of the architecture, Coordinator Module receives document from crawler as parameter and performs the following steps:

i) Identify document domain and format
ii) Decide memory block
iii) Extract URLs
iv) Add URLs to seed of URLs
v) Compress the document
vi) Update Index
vii) store compressed document

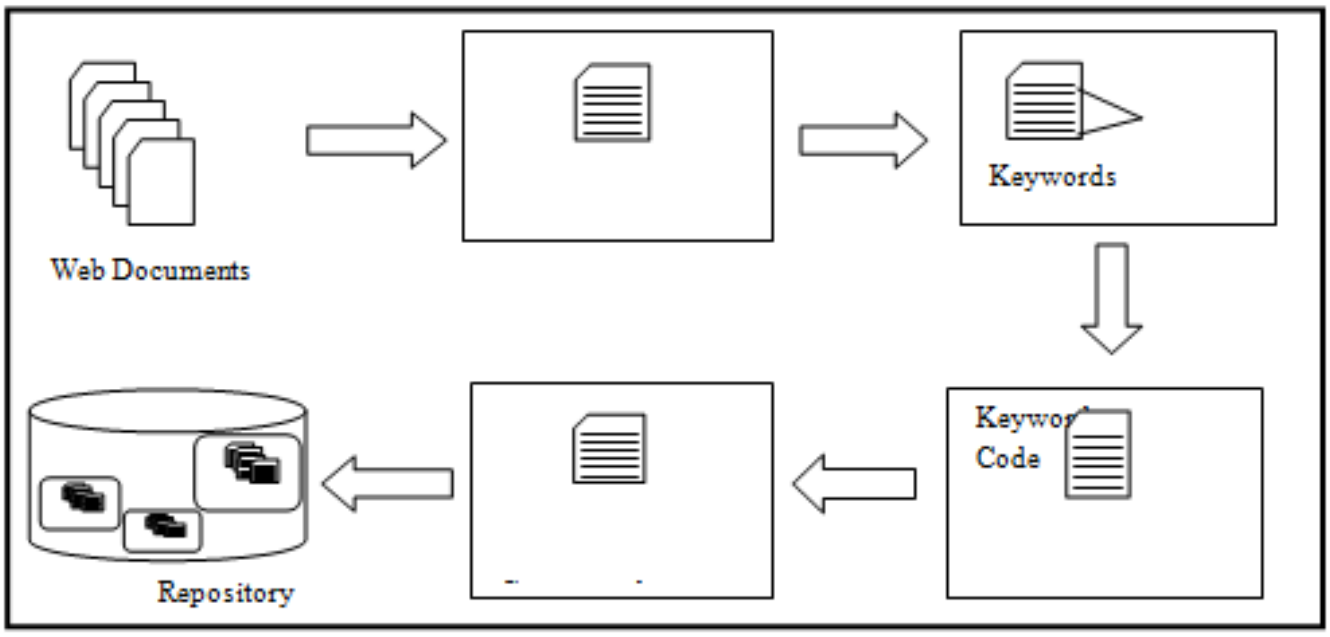

Figure.3. Working of Coordinator Module

On receiving the document from the crawler, coordinator module identifies the format and domain of downloaded document. On the basis of domain of that document it decides the cluster to check whether the document is already downloaded or not using URL identifier. Thus reducing the search complexity because searching is done only in a specific cluster not in linear fashion to complete repository. If the document is fresh one then, coordinator module checks the status of cluster that the cluster can acquire new document or not. If the status of cluster is able to have new document then extract URLs, add them to seed of URLs, the document is parsed and the keywords are identified along with the frequency. Coordinator receives frequency from indexer to generate the Huffman codes using Huffman algorithm. Table 4.1 shows structure of indexer. In the proposed architecture, frequency depends on the presence of keyword to different documents. The keyword that is present in more number of documents will have higher frequency that leads to less length size Huffman code according to Huffman algorithm. Finally the index is updated and the document is saved in compressed form or else the status of cluster is not able to accommodate new document than less ranked document will be deleted to store new document. Figure 4.3 depicts how the coordinator module deals with web pages and stores them to repository in compressed form using Huffman algorithm concept.

The indexer supports inverted indexing technique. The structure of indexer has four basic information:

- keyword
- Domain Name
- Frequency
- Document Names

Table .1. Structure of Index

Keyword	Domain Name(Cluster)	Frequency	Document Name
Crawler	.org	9	Document 1, Document 3, Document 4, Document 7
	.com		Document 2, Document 12
	.co		Document 6, Document 8, Document 10
Network	.in	3	Document 11, Document 5
	.edu		Document 19
……	……..	…….	……..

In the case when document already exists then ranking module decide the importance of the document. Algorithm of Coordinator module:

```
Coordinator_module (domain, document)
{
        Steps:
        1.Identify the document domain like doc, html, or pdf etc.;
        2.Check the domain of downloaded document , decide the memory block  and search it only its
          specific domain cluster (memory block)whether the document has already downloaded or not;
        3.If(document is fresh one)

          {
              3.1 extract the links or references to the other cites from that documents;
              3.2 Compress the document using Huffman algorithm;
              3.2 Update Index;
              3.2 Check the status of decided cluster and if there is a need to create space for new
                  downloaded document then remove less ranked document;
              3.3 Store compressed document it into repository;
          }
        4. else
          {
              4.1 Call ranking_module (cluster, document);
              4.2 Convert the URL links into their absolute URL equivalents;
              4.3 Add the URLs to set of seed URLs;
          }
}
```

Figure.4. Algorithm of Coordinator Module

Functioning of Ranker Module

This component retrieves cluster and document from coordinator module, and increases the rank of document based on popularity to maintain the freshness of repository also help coordinator module to maintain the size of clusters. It calculates rank of document on the basis of popularity, increase rank and return rank. The main steps of Ranker module:

```
Ranking_module (cluster, document)
{
        Steps:
        1. Calculate rank of document on the basis of popularity;
        2 .Increase rank;
        3. Return rank;
}
```

Figure.5. Algorithm of Ranker Module

3. CONCLUSION

The proposed work presented clustered web repository, and meta-data management. Clustered Web base uses repository and working modules to distribute data among domain specific clusters which compose a clustered web repository. Distribution of web pages to different domain specific blocks reduces searching complexity. The proposed work also representing an indexing mechanism to store the keyword present in the document in compressed form by mapping variable length Huffman code. It also focuses on the presence of keyword in different document because the keyword in maximum number of documents will be mapped to less length Huffman code. The mechanism reduces the size of document and after updates the index. The data structure of the indexer fastens the search for matched results from the Inverted Index with the cluster information. It also helps the user to process the user query with fast and more relevant results.

4. REFERENCES

[1]. Gary C. Kessler, “An Overview of TCP/IP Protocols and Internet”,2007.

[2]. Grossan, B, “Search Engines : What they are, how they work, and practical sugg estions for getting the most out of them”, 1997.

[3]. A. Ntoulas, J. Cho, and C. Olston, “What's new on the web ? : the evolution of the web from a search engine perspective.”, In Proceedings of the *13th conference on World Wide Web*, pages 1-12, 2004.

[4]. Arvind Arasu, Junghoo Cho, Hector Garcia-Molina, Andreas Paepcke, Sriram Raghavan, "Searching the Web.", *ACM Transactions on Internet Technology*, 2001.

[5]. Brewington, B. E., Cybenko, G, “Keeping up with the changing web”, *IEEE Computer*, 2000.

[6] Komal Kumar Bhatia, A. K. Sharma, “A Framework for Domain-Specific Interface Mapper (DSIM)”, *International Journal of Computer Science and Network Security* (IJCSNS), Vol 8, No12, Dec 2008.

[7]. Niraj Singhal, Ashutosh Dixit, "Retrieving Information from the Web and Search Engine Application", in proceeding of *National conference on "Emerging trends in Software and Networking Technologies ETSNT'09*", Amity University, Noida, India, pp. 289-292,2009.

[8]. S. Chakrabarti, M. van den Berg, and B. Dom., "Focussed Crawling: a New Approach to Topic-specific Web Resource Discovery", In Proceedings of the *Eight International World-Wide Web Conference*, pages 545-562, Toronto, Canada, May 1999.

[9]. Niraj Singhal, Ashutosh Dixit, "Web Crawling Techniques : A Review", in proceedings of *National Seminar on "Information Security : Emerging Threats and Innovations in 21st Century*", Ghaziabad, March 2009, pp. 34-43.

[10]. A. Heydon, Najork M., "Mercator : A scalable, extensible Web crawler.", *World Wide Web*, vol. 2, no. 4, pp. 219-229, 1999.

[11]. Berners-Lee and Daniel Connolly, "Hypertext Markup Language. Internetworking draft", http:// www.w 3.org/ hypertext/WWW/MarkUp/ HTML.html, 13 Jul 1993.

[12]. B. Kahle,"Archiving the Internet", *Scientific American*, 1996.

[13]. M. Gray, "Measuring the growth of the Web", http://www.mit.edu /people/mkgray/growth/, 1993.

[14]. Barry M. Leiner, Vinton G. Cerf, David D. Clark, Robert E. Kahn, Leonard, Kleinrock, Daniel C. Lynch, Jon Postel, Larry G. Roberts, Stephen Wolff, "A Brief History of the Internet", www.isoc.org/internet/history.

[15]. Bernardo A. Huberman, Lada A. Adamic, "Growth dynamics of the World-Wide Web. Nature",1999.