Web Cache Optimization in Semantic based Web Search Engine

Dr.S.N.Sivanandam
Advisor,Akshya College of Engineering, Coimbatore., India.

Dr.M.Rajaram
Professor & Head, Government College of Engineering, Tirunelveli, India

S.Latha Shanmuga Vadivu,
Assistant Professor, ECE Department, Tamilnadu College of Engineering, Coimbatore, India.

ABSTRACT
With the tremendous growth of information available to end users through the Web, search engines come to play ever a more critical role. Nevertheless, because of their general-purpose approach, it is always less uncommon that obtained result sets provide a burden of useless pages. The next-generation Web architecture, represented by the Semantic Web, provides the layered architecture possibly allowing overcoming this limitation. The ontology for multiple search engines is written such that in this search engine for single query the final result is got from multiple search engines. After getting the user query result we can use the clustering. In this clustering the user query results is formed in the a to z form, the several search engines have been proposed, which allow increasing information retrieval accuracy by exploiting a key content of Semantic Web resources, that is, relations. We can use web cache optimization in search engine to get fast retrieval of user query results. In this work I have used web cache optimization based on eviction method for semantic web search engine. In this paper, analization of both advantages and disadvantages of some current Web cache replacement algorithms including lowest relative value algorithm, least weighted usage algorithm and least unified-value algorithm is done. Based on our analysis, we proposed a new algorithm, called least grade replacement (LGR), which takes recency, frequency, perfect-history, and document size into account for Web cache optimization**.**

KEYWORDS*:*
Semantic web, multiple search engines, ontology, clustering, web caching, LRU, LGR, LUV algorithm.

1. INTRODUCTION
The Semantic Web is known for being a web of Semantic Web documents; however, little is known about the structure or growth of such a web. Search engines such as Google have transformed the way people access and use the web and have become a critical technology for finding and delivering information [1]. Most existing search engines, however, provide poor support to accessing the web of result's and make no attempt to take advantage of the structural and semantic information encoded in SWDs. The Semantic Web will offer the way for solving this problem at the architecture level. In fact, in the Semantic Web, each page possesses semantic metadata that record additional details concerning the Web page itself. This work is designed to serve the research activities in Semantic Web community, especially the following:

(i) Multiple Search Engine for single user query
(ii) Apply Clustering Method
(iii) Unwanted pages in the result set would force him or her to perform a post processing on retrieved information to discard unneeded ones. Today, search engines constitute the most helpful tools for organizing information and extracting knowledge from the Web. However, it is not uncommon that even the most renowned search engines return result sets including many pages that are definitely useless for the user this is mainly due to the fact that the very basic relevance criterions underlying their information retrieval strategies rely on the presence of query keywords within the returned pages. When a user enters a query composed by the following keywords "hotel," "Rome," and "historical center" (or "hotel," "Roma," and "centrostorico") in the Italian version of the well-known Google search engine [7]. He or she would not be astonished probably by finding that the result set actually includes several hotels located in the historical center of Rome, as expected small town at some distance from the Rome city center is also included. However, two hotels located in the historical center of other main Italian cities are also displayed. Finally, three hotels named Roma are included among the 10 most relevant results even if they have nothing to do with the selected city. Only 4 out the 10 results presented to the user satisfy user needs. (Even if they seem to satisfy the user query, based on the strategy adopted to process it). Currently, the Semantic Web, (i.e. online documents written in RDF or OWL), is essentially a web universe parallel to the web of HTML documents. Semantic Web documents (SWDs) are characterized by semantic annotation and meaningful references to other SWDs [5]. Since conventional search engines do not take advantage of these features, a search engine customized for SWDs, especially for ontology's, is needed by human users as well as by software agents and services[3]. At this stage, human users are expected to be semantic web researchers and developers who are interested in accessing, exploring and querying RDF and OWL documents found on the web.

2. WEB CACHE DESIGN
We discuss three general purpose cache distributions and lookup enhancements that improve both the locality and latency of cache advertisements. The system uses a form of hierarchical aggregation to summarize the contents of cached files available in a particular local area network. In this way the amount of indexing information that has to be exported to other systems in a WAN can be reduced. A common criticism of distributed hash tables is that they lack locality. This is a side effect of the hash function used to identify both nodes and content in the DHT network. The hash function provides a key-to node mapping that distributes keys uniformly at random across the address space of the DHT [10]. As such, semantically related nodes and data items when processed by a systems hash function will be mapped to random locations in the network with high probability. This presents a number of problems for cache index and lookup systems. First, lookup requests for file content such as images and linked web pages require a separate lookup request for each URL. This will result in a worst-case time complexity of O (M log N) where M is the number of embedded file references in a webpage and N is the number of nodes in the system. Second, due to the random nature of the hash functions used to identify files, lookup requests for linked files are likely to be routed to nodes that are far away in the network. This significantly adds to the latency of locating a cached file in the network. However, many of these lookup requests are unnecessary and can be reduced by exploiting the link

structure of web pages. In a typical web browsing scenario, client Software will make a connection to a web server and download the HTML specification of a web document. Once this has occurred, the client process will parse the document tree and generate a series of HTTP get requests to download embedded file content from the web server. As such, this content should also be available at the same remote cache system as the main webpage unless it has been evicted from the remote cache. To reduce these extraneous lookup requests, cache misses and extra round-trip delays, we have developed a combined indexing structure that client systems can use to identify the set of related cache items also available at a remote site. This combined index has been implemented using a bitmap vector the contents of which are used to determine the presence or absence of linked web content. This effectively allows a client system to select a remote cache based upon the number of related documents that it stores. As a result, lookup requests for related files such as embedded images can be downloaded from the same remote cache without having to specifically locate the file using the DHT index. The idea here is to extend the reach of the cache index by one link's worth, to enable a client system to determine ahead of time whether linked content is available at a remote proxy. As a consequence, communication between a client and remote cache system can be reduced because of these cache hints. This allows a client system to maintain a persistent connection with a remote cache, so that file requests for linked web content can be pipelined across the same socket. To create this combined index, the link structure of a cached file has to be extracted using regular expressions. This process creates an ordered set of links that can be used to create a bitmap vector of the linked files available at a remote site. As such, the length of a bitmap vector corresponds to the number of out links in a given web page[12]. To encode the availability of linked content at a remote site, the corresponding bit locations of these out links are set in the bitmap. Therefore, the ith link is represented by the ith bit in the bitmap vector. To illustrate this idea, consider a web page that has five links to other files. If each of these linked files were available at a remote cache, then each bit location in the bitmap vector of this cached item would be set to one. However, if only the second and third links were available at a remote cache, then only bit locations one and two would be set in the bitmap. The intuition here is that users will browse to a new page through an existing hyperlink directly, instead of jumping to a new page at random[11].Therefore, if we know which links are available ahead of time, the number of cache lookup messages routed across the network can be reduced. Once a browser has downloaded a list of IP addresses and adjacency cache bitmaps from the DHT, these are added to a fixed size in memory cache which has a least recently used eviction strategy.

2.1 EXISTING SYSTEM

Nevertheless, because of their general-purpose approach, it is always less uncommon that obtained result sets provide a burden of useless pages. It is not uncommon that even the most renowned search engines return result sets including many pages that are definitely useless for the user this is mainly due to the fact that the very basic relevance criterions underlying their information retrieval strategies rely on the presence of query keywords within the returned pages [10].

1. Lowest Relative Value Algorithm (LRV)

Luigi and Vicisano proposed a replace algorithm for proxy cache called Lowest Relative Value (LRV). It is based on maximizing an objective function for the whole cache. The objective function uses a cost/benefit model to calculate the relative value of each document in the cache. Two performance parameters of cache are used: the HR and BHR [11,12].

2. Least Weighted Usage Algorithm (LWU)

Ying, Edward, and Ye-sho argued that model-driven simulation was more objective than trace-driven. A web cache algorithm called Least Weighted Usage (LWU) was proposed using model-driven simulation [11,12].

3. Least Unified Value Algorithm (LUV)

Bahn et al. proposed a web cache replacement algorithm called LUV that uses complete reference history of documents, in terms of reference frequency and recency [11,12].

Disadvantage of Existing System

(i) Text based searching example (Google, yahoo, msn, Wikipedia).
(ii) Without semantic relationship to give exact result.
(iii) Query only focus single search engine.
(iv) Most existing search engines however, provide poor support to accessing the web results.
(v) No analysis of stopping keywords from the user query.
(vi) It will not give relevant or exact result.
(vii) Number of iterations is high.
(viii) A replacement policy is required for replacing a page from web cache to make room for new page.

2.2 PROPOSED SYSTEM

The Semantic Web will offer the way for solving this problem at the architecture level. In fact, in the Semantic Web, each page possesses semantic metadata that record additional details concerning the Web page itself. It will be proved that relations among concepts embedded into semantic annotations can be effectively exploited to define a ranking strategy for Semantic Web search engines. A similarity score measuring the distance between the systematic descriptions of both query and retrieved resources is defined. They first explode an initial set of relations (properties) by adding hidden relations, which can be inferred from the query. Similarity is then computed as the ratio between relation instances linking concepts specified in the user query and actual multiplicities of relation instances in the semantic knowledge base. This method is applied on each property individually and requires exploring all the Semantic Web instances. Moreover, the user is requested to specify all the relations of interest. Thus, since it is predictable that the number of relations will largely exceed the number of concepts, its Applicability in real contexts is severely compromised. A similar approach, aimed at measuring the relevance of a semantic association (that is, a path traversing several concepts linked by semantic relations) [3]. We provide an interesting definition of relevance as the reciprocal of the ambiguity of the association itself. Ontology-based lexical relations like synonyms, antonyms, and homonyms between keywords (but not concepts) have been used to "expand" query results which automatically associate related concepts, and exploit the semantic knowledge base to automatically formulate formal queries.

This work web cache (proxy server) is to develop a utility to share internet from single connection to a large network around 200 machines with different operating systems. The software is developed using the Java Language. Java applet applications are mostly used in the web pages, but we use JFC (swing) for developing the software [10].

This work provides an intelligent environment containing a number of ready-made options like cache, log file, error checking, connection pooling, etc. These ready-made tools may be any of the GUI components that are available in the Java AWT package. By using this utility, Administrator can control and maintain the whole network. This thesis aim is to use the Least Recently Used document in web caches which replaces Randomized web cache replacement algorithm. A web cache sits between web server and a client and watches request for web

pages. It caches web documents for serving previously retrieved pages when it receives a request for them.

2.3 SEMANTIC WEB INFRASTRUCTURE

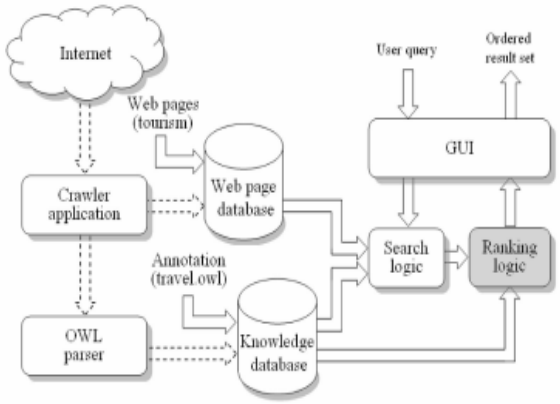

Fig 1: ArchitectureDiagram.

In Fig1.the crawler application collects annotated Web pages from the Semantic Web (in this case, represented by the controlled environment and its Web page collection) including RDF metadata and originating OWL ontology. RDF metadata are interpreted by the OWL parser and stored in the knowledge database. A graphics user interface allows for the definition of a query, which is passed on to the relation-based search logic. The ordered result set generated by this latter module is finally presented to the user. The details of the system workflow will be provided in the following sections.

2.4 ADVANTAGE OF PROPOSED SYSTEM

The Semantic Web will offer the way for solving this problem at the architecture level. In fact, in the Semantic Web, each page possesses semantic metadata that record additional details concerning the Web page itself. This method is applied on each property individually and requires exploring all the Semantic Web instances.

The ontology for multiple search engines is written such that in this search engine for single query the final result is got from multiple search engines. After getting the user query result we can use the clustering. In this clustering the user query results is formed in the a to z form. The Several search engines have been proposed, which allow increasing information retrieval accuracy by exploiting a key content of Semantic Web resources, that is, relations reduces network traffic, reduces Latency time, to reduce load on web servers and avoid the need for data structures.

3. IMPLEMENTATION

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective. The implementation stage involves careful planning, investigation of the existing system and it's constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods. The implementation of this work is done using Java and run by means of NETBEANS IDE 6.8 platform.

3.1 Web Search Engine Design

The term "search engine" is often used generically to describe both crawler-based search engines and human-powered directories. These two types of search engines gather their listings in radically different ways. Crawler-based search engines, such as Google, create their listings automatically [2]. They "crawl" or "spider" the web, then people search through what they have found. A human-powered directory, such as the Open Directory, depends on humans for its listings. When we submit a short description to the directory for your entire site or editors write one for sites they review. A search looks for matches only in the descriptions submitted.

3.1.1 Web Crawler

A search engine cannot work without a proper index where possible searched pages are stored, usually in a compressed format. This index is created by specialized robots, which crawl the Web for new/modified pages (the actual crawlers, or spiders)[4]. Typical crawler architecture is depicted in the figure2 below.

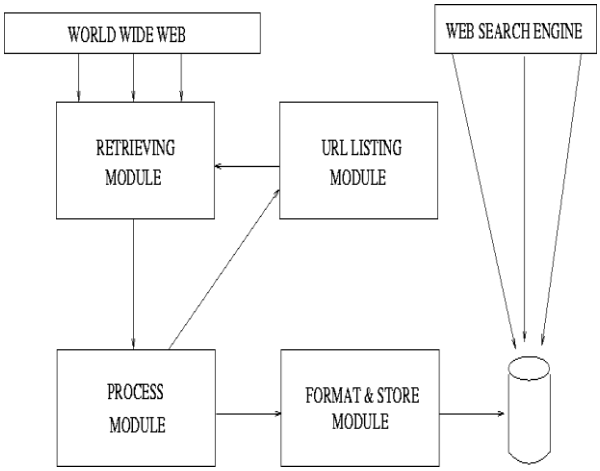

Fig 2: Typical crawler Architecture

3.1.2 Multiple Search engine Design

The most known general search engines are Google and Yahoo! but one of the oldest search engines is AltaVista. All existing search engines have weaknesses, even Google (link searches must be exact, it does not support full Boolean, it only indexes a part of the web pages or PDF files, etc.). This part represents a real reason for building more search engine. A scalable distributed repository is used to store the crawled collection of Web pages. Strategies for physical organization of pages on the storage devices, distribution of pages across machines, and mechanisms to integrate freshly crawled pages, are important issues in the design of this repository. The repository supports both random and stream-based access modes. Random access allows individual pages to be retrieved based on an internal page identifier. Stream-based access allows all or a significant subset of pages to be retrieved as a stream. Query-based access to the pages and the computed features (from the feature repository) is provided via the Web Base query engine[8].Unlike the traditional keyword-based queries supported by existing search engines, queries to the Web Base query engine can involve predicates on both the content and link structure of the Web pages.

In selection of search engines twenty five search engines were selected to conduct our experiment. They are AlltheWeb, AltaVista, google, yahoo, clusty, you tube, file tube,

citeceer etc., to name a few. At first, the search engines were selected and the user query is submitted to all search engines under consideration. The queries covered a broad range of topics. The topics are as follows: Computer science, education, Internet, literature, music, plants, sports, travel etc. The precision of content of these pages is compared to give the result.

3.1.3 Design of Ontology Search

As mentioned in the last section, finding ontologies to satisfy user requirements is a very important issue, in both KB reuse and Semantic Web areas. There is no existing tool to solve this problem. Google does have the power, but does not seem to be specific enough to give good results [5]. After some experiments, we noticed that the problem arises because Google does not offer a good visualization function for the ontology files (in different formalisms, such as RDFs, etc.), as the user cannot view the ontology in an intuitive graphic format; they have to look thro ugh the ontologies as structured text files. This process takes a lot of time and cannot guarantee a good result, as the plain text of the ontology cannot show the internal structure of the ontology clearly. The ontology searching steps is shown in the figure 3 below.

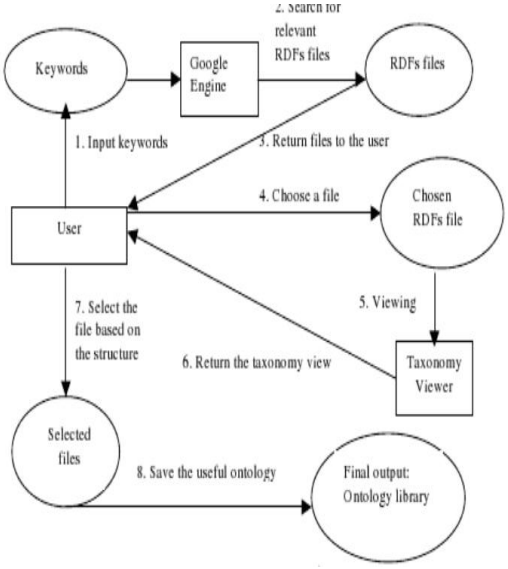

Fig 3: Ontology searching steps

3.1.4 Clustering the web result's

Clustering is the act of grouping similar object into sets. In the web search context: organizing web pages (search results) into groups, so that different groups correspond to different user needs. Existing search engines such as Google and Yahoo return ranked lists of Web pages in response to a user's query request. Web users have to shift through the list to locate pages of interest [9]. This is a time-consuming task when multiple sub-topics of the given query are mixed together. A possible solution to this problem is to cluster search results into different groups and to enable users to identify their required group at a glance.

4. PROXY IMPLEMENTATION

We have developed a PROXY SERVER, which runs with mentioned features, which inherently helps speeder browsing of web pages with use of least grade page replacement algorithms. This server is successfully implemented with a few numbers of clients but it could be implemented for more of them. As mentioned before it is more reliable, more advantageous than the existing one which uses the old Data structures concept. It can work in a larger network and also maintains load balancing so I conclude that this system application is executable under any platform and with any number of clients too.

4.1 HTTP CONTENT AND PARSING ANALYSIS

Parsing is the process of analyzing an input sequence in order to determine its grammatical structure with respect to a given formal grammar [7].

PROCESS STEPS

4.1.1Lexical analysis*:*

The input character stream is split into meaningful symbols (tokens) defined by a grammar of regular expressions. Example: the lexical analyzer takes "12*(3+4)^2" and splits it into the tokens 12, *, (, 3, +, 4,), ^ and 2.

4.1.2 Syntax analysis

It performs checking if the tokens form an legal expression, with respect to a CF grammar. Limitations are it cannot check (in a programming language): types or proper declaration of identifiers

4.1.3 Semantic parsing

Works out the implications of the expression validated and takes the appropriate actions. Examples: an interpreter will evaluate the expression, a compiler, will generate code.

4.2 EXPERIMENTAL RESULTS

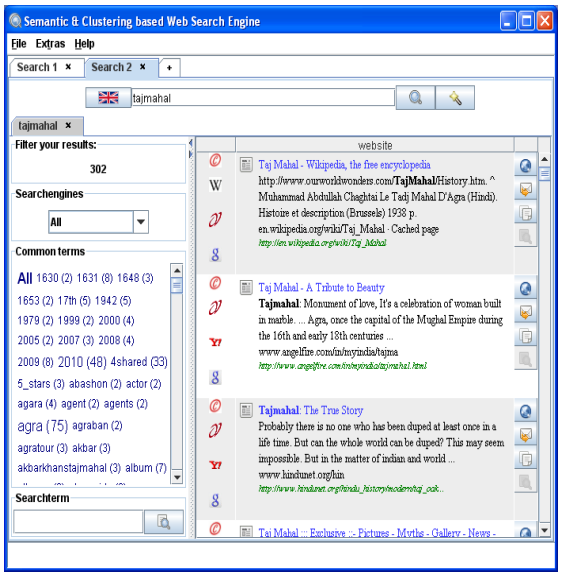

Fig 4: Result Window for the query Tajmahal

The lexical parser and syntax parser supports for forming the filter value of the user query. The common terms are clustered based on the user query given. The clustering concept used here is k means clustering algorithm. The user query takes the relation of some common terms and gives the result as shown in figure 4. For a single user query Tajmahal the result is got from multiple search

engines; such as Google, Yahoo, AltaVista, Clusty, Excite, All the web, File tube, You tube, Amazon, Cite seer, Wikipedia, Isohunt etc., here the answer is got from 25 search engines, searches take place simultaneously from different search engines, the main advantage is that searches do not have to wait for each search engine, the results are computed simultaneously from multiple search engines and they are displayed on the screen. The result is therefore much faster.

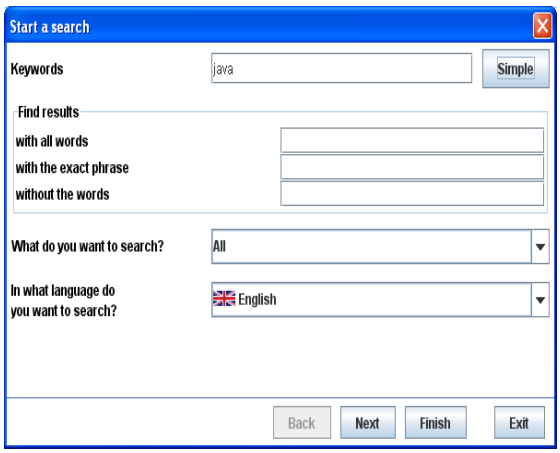

Fig 5: Advanced Query Search

This is advanced window search processing. It can find the result with all words, exact phrase and without word's.

TABLE 1: PERFORMANCE COMPARISON

Web search engine URL	User query	Normal query result	Web cache optimization result	Semantic based web result
www.google.com	what is java swing	50	80	95
www.yahoo.com	what is java swing	30	60	79
www.altavista.com	what is java swing	60	75.5	90.1
www.youtube.com	what is java swing	10	60	89.5
www.clusty.com	what is java swing	60	80	98.9
www.filetube.com	what is java swing	59	75	88
www.torrentz.com	what is java swing	70	50	80
www.citeseer.com	what is java swing	70	78	95

The comparison table shows search engine performance for the user query what is java swing? That user query gets the result from different search engines. I can measure the user query based on normal query result, without any semantic analysis and web cache optimization analysis. Then the same query get different result from different search engine using web cache optimization. Then the same user query get the different search engine result based on semantic web result. Then I can form the bar chart using three column values. In a similar way a line chart is formed for the same query with three column values to analyze the performance.

5. FETCH UNITS AND RESULT PRE-FETCHING

In many search engine architectures, the computations required during query execution are not greatly affected by the number of results that are to be prepared, as long as that number is relatively small. In particular, it may be that for typical queries, the work required to fetch several dozen results is just marginally larger than the work required for fetching 10 results. Since fetching more results than requested may be relatively cheap, the dilemma is whether storing the extra results in the cache (at the expense of evicting previously stored results) is worthwhile. Roughly speaking, result prefetching is profitable if, with high enough probability, those results will be requested shortly - while they are still cached and before the evicted results are requested again. One aspect of result prefetching was analyzed in, where the computations required for query executions (and not cache hit ratios) were optimized [10].

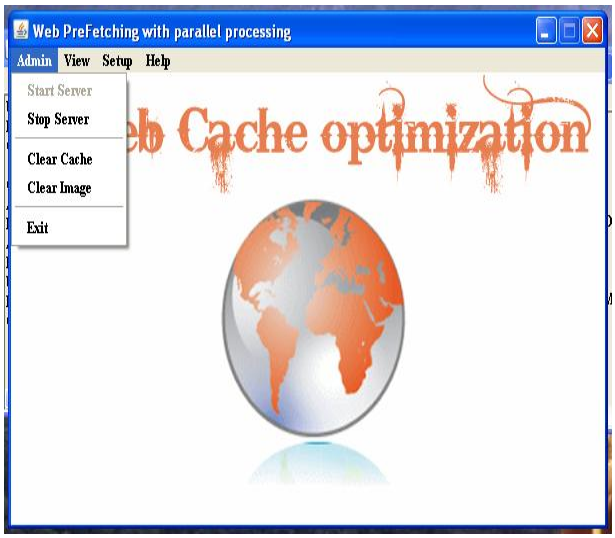

Fig 6: Web cache main GUI showing Admin Menu

The Figure 6 shows the web cache main GUI. Here java swing is used to build the user interface. It has four menus, Admin menu, View menu, Setup menu and Help menu.

The Admin menu is shown in figure. It has four submenus. They are Start server to start HTTP server, Stop server to stop the HTTP server, Clear cache clears manually all the cache files in your cache directory, Clear image manually clears all the images in your image directory.

The View menu has four submenus, Server response to get the server log Information and also monitoring the connection between server and internet you can save log file for future reference, Client request monitors the connection between the local server and number of clients. This process monitors every client request response and also identifies which ask the web URL's and also save log information for future reference. The picture viewer is used to view all the pictures in your cached picture directory. The window is used to see each single image size and dimension and also to delete the unwanted single image from the cached image directory.

Cache view is used to view the cache file from the cached directory and then view file and save the cached file in different formats, for example html, jsp and asp.

The Setup menu gives information about the server configuration which has two Submenus, The network input configuration window were we can set the proxy input and port number and the connection properties can be set like receive

timeout, send time out number log of threads, cached url's and cache expires (hrs) time. It is used for refreshing the server every one hour using N-M algorithm in LRU method.The logging window gives the access log and error log information. The access log caches the errors based on pages got from URL. The error log shows connection error between client and server. The logging level is based on minimal normal and detail information about the error. The Help menu shows the information about the server.

6. CONCLUSION

The next-generation Web architecture represented by the Semantic Web will provide adequate instruments for improving search strategies and enhance the probability of seeing the user query satisfied without requiring tiresome manual refinement. Nevertheless, they mainly use page relevance criteria based on information that has to be derived from the whole knowledge base, making their application often unfeasible in huge semantic environments. By neglecting the contribution of the remaining annotated resources, a reduction in the cost of the query answering phase could be expected. Despite the promising results in terms of both time complexity and accuracy, further efforts will be requested to foster scalability into future Semantic Web repositories based on multiple ontology, characterized by billions of pages, and possibly altered through next generation "semantic" spam techniques. It has been designed and partially implemented to capture more metadata on classes and properties and to support millions of documents. We have also built an ontology dictionary based on the ontologies discovered by our research, which we continue to refine. We have described a prototype crawler-based indexing and retrieval system for Semantic Web documents. The traditional browsing cache systems can not address both non stationary and stationary browsing behaviors at the same time. The response time for an interactive browsing system can be greatly increased.

7. REFERENCES

[1] B.Aleman-Meza, C. Halaschek, I. Arpinar, and A. Sheth, "A Context-Aware Semantic Association Ranking," Proc. First Int'l Workshop Semantic Web and Databases (SWDB '03), pp. 33-50, 2003.

[2] K. Anyanwu, A. Maduko, and A. Sheth, "SemRank: Ranking Complex Relation Search Results on the Semantic Web," Proc. 14th Int'l Conf. World Wide Web (WWW '05), pp. 117-127, 2005.

[3] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," scientific Am., 2001.

[4] S. Brin and L. Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," Proc. Seventh Int'l Conf. World Wide Web (WWW '98), pp. 107-117, 1998.

[5] Seda Cakiroglu, Erdal Arikan, "Replace Problem in Web Caching", in Proceedings of IEEE Symposium on Computers and Communications, June, 2003.

[6] S. Cohen, J. Mamou, Y. Kanza, and Y. Sagiv, "XSEarch: A Semantic Search Engine for XML," Proc. 29th Int'l Conf. Very Large Data Bases, pp. 45-56, 2003.

[7] Berners-lee, t., Hendler, j., and lassila, o. (2001) "The Semantic Web". Scientific American, May http://www.sciam.com/2001/0501issue/0501berners-lee.html

[8] L. Ding, T. Finin, A. Joshi, R. Pan, R.S. Cost, Y. Peng, P. Reddivari, V. Doshi, and J. Sachs, "Swoogle: A Search and Metadata Engine for the Semantic Web," Proc. 13th ACM Int'l Conf. Information and Knowledge Management (CIKM '04), pp. 652-659, 2004.

[9] L. Ding, T. Finin, A. Joshi, Y. Peng, R. Pan, and P. Reddivari, "Search on the Semantic Web," Computer, vol. 38, no. 10, pp. 62-69, Oct. 2005.

[10] H. Bahn, S. Noh, S. L. Min, and K. Koh, "Using Full Reference History for Efficient Document Replacement in Web Caches", in Proceedings of the 2nd USENIX Symposium on Internet Technologies & Systems, October, 1999.

[11] Ying Shi, Edward Watson, and Ye-sho Chen, "Model-Driven Simulation of World-Wide-Web Cache Policies", In Proceeding of the *1997* Winter Simulation Conference, June, 1997.

[12] Ganesh, Santhanakrishnan, Ahmed, Amer, Panos K. Chrysanthis and Dan Li, "GDGhOST: A Goal Oriented Self Tuning Caching Algorithm", in Proceeding of the 19th ACM Symposium on AppliedComputing.March, 2005.