International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 1 Issue 4, June - 2012

Improving Response Time and Throughput of Search Engine with Web Caching

Pushpa C N[1], Thriveni J[1], Venugopal K R[1], Patnaik L M[2]
[1]*Department of Computer Science and Engineering,*
University Visvesvaraya College of Engineering,
Bangalore- 560001
[2] Honorary Professor, Indian Institute of Science, Bangalore, India.

Abstract

Large web search engines need to be able to process thousands of queries per second on collections of billions of web pages. As a result, query processing is a major performance bottleneck and cost factor in current search engines, and a number of techniques are employed to increase query throughput, including massively parallel processing, index compression, early termination, and caching. Caching is a useful technique for Web systems that are accessed by a large number of users. It enables a shorter average response time, it reduces the workload on back-end servers, and it reduces the overall amount of utilized bandwidth. Our contribution in this paper can be split into two parts. In the first part, we proposed Cached Search Algorithm (CSA) on top of the multiple search engines like Google, Yahoo and Bing and achieved the better response time while accessing the resulting web pages. In the second part, we design and implemented the Cached Search Engine and the performance evaluated based on the training data (WEPS dataset [1]) and the test data (Mobile dataset). The Cached Search outperforms the better by reducing the response time of search engine and to increase response throughput of the searched results.

1. Introduction

The Internet and the Web offer new opportunities and challenges to information retrieval researchers. With the information explosion and never ending increase of web pages as well as digital data, it is very hard to retrieval useful and reliable information from the Web.

Search Engines have become a popular way of finding information on the World Wide Web. A large number of users reach web sites via a Search Engine. As the web become larger, individual sites become more difficult to find, and thus the percentage of users that reach web sites after querying a Search Engine will probably increase.

The basic functions of a crawl-based web search engine can be divided into four stages: data acquisition (or crawling), data mining and preprocessing, index construction, and query processing. During crawling, pages are fetched from the web at high speed, either continuously or through a set of discrete crawls. Then various data mining and preprocessing operations are performed on the data, e.g., detection of web spam or duplicates, or link analysis based on PageRank [2]. Third, a text index structure is built on the preprocessed data to support fast query processing. Finally, when a user issues a query, the top results for the query are retrieved by accessing the index structure.

1.1 Web Caching

Caching is a useful technique for Web systems that are accessed by a large number of users. It enables a shorter average response time, it reduces the workload on back-end servers, and it reduces the overall amount of utilized bandwidth. In a Web system, both clients and servers can cache items. Browsers cache Web objects on the client side, whereas servers cache pre-computed answers or partial data used in the computation of new answers.

Web caching is the caching of web documents , such as HTML pages and images, to reduce bandwidth usage, server load, and perceived lag. A web cache stores copies of documents passing through it; subsequent requests may be satisfied from the cache if certain conditions are met. Web cache optimization in search engine is used to get fast retrieval of user query results. Web objects can be cached locally on the user's computer or on a server on the Web. One such optimization is the use of caching, which occurs in search engines on two levels. A query enters the search

engine via a *query integrator* node that is in charge of forwarding it to a number of machines and then combining the results returned by those machines. Before this is done, however, a lookup is performed into a cache of previously issued queries and their results. Thus, if the same query has been recently issued, by the same or another user, then we do not have to recompute the entire query but can simply return the cached result. This approach, called *result caching*, is widely used in current engines. A second form of caching, called *index caching* or *list caching* [3], is used on a lower level in each participating machine to keep the inverted lists of frequently used search terms in main memory. Our main focus is on result caching.

1.2 Types of Web Caching

There are several types of caches for web objects:

- **Browser cache:** Browsers' cache Web objects on the user's machine. A browser first looks for objects in its cache before requesting them from the website. Caching frequently used Web objects speeds up Web surfing. For example, we often use Google.com and yahoo.com. If their logos and navigation bars are stored in the browser's cache, then the browser will pick them up from the cache and will not have to get them from the respective websites. Getting the objects from the cache is much faster than getting them from the websites.
- **Proxy cache:** A proxy cache is installed near the Web users, say within an enterprise. Users in the enterprise are told to configure their browsers to use the proxy. Requests for objects from a website are intercepted and handled by the proxy cache. If they are not in the cache, the proxy gets them from another cache or from the website itself.

1.3 Advantages of Web Caching

Web Caching has the advantages like

i) Faster delivery of Web objects to the end user.
ii) It reduces bandwidth cost and needs.
iii) It benefits the user, the service provider and the Website owner.
iv) It reduces load on the website servers.

Motivation: The growing sophistication of search engine software enables us to precisely describe the information that we seek. Millions of queries are submitted daily to Web search engines, and users have high expectations of the quality of results and the latency to receive them. As the searchable Web becomes larger, with more than 20 billion pages to index, evaluating a single query requires processing large amounts of data. In such a setting, using a cache is crucial to reduce the response time and to increase the response throughput.

Contribution: Our contribution in this paper can be split into two parts. In the first part, we proposed Cached Search Algorithm (CSA) on top of the multiple search engines like Google, Yahoo and Bing and achieved the better response time while accessing the resulting web pages. In the second part, we design and implemented the Cached Search Engine and the performance of this is evaluated based on the training data (WEPS dataset [1]) and the test data (Mobile dataset). The Cached Search outperforms the better by reducing the response time of search engine and to increase response throughput of the searched results.

Organization: The remainder of the paper is organized as follows: Section 2 reviews the related work of the web search engine with web caching, Section 3 explains the system architecture of web caching for multiple search engines, Section 4 gives the problem definition and the Cached Search Algorithm. In Section 5 we explained the implementation of the Cached Search engine, the Section 6 describes the web services used, Section 7 gives the Performance Analysis and results and Conclusions are presented in Section 8.

2. Related Work

Junghoo Cho et. al. [4] proposed a method for Efficient Crawling through URL ordering. A crawler is a program that retrieves web pages, commonly for use by a search engine or web cache. Different metrics are defined and three models to evaluate crawlers. Evaluated experimentally several combinations of importance and ordering metrics, using Stanford web pages. Drawback of this is, it run only over the Stanford web pages. Future work proposed here is working over non-Stanford web pages to analyze structural differences and their implication for crawling.

Jon M. Kleinberg et. al. [5] developed a set of algorithmic tools for extracting information from the link structures of Hyperlinked environment. It is a technique for locating high-quality information related to a broad search topic on the www, based on a structural analysis of the link topology surrounding *authoritative* pages on topic. Francois Bry et. al. [6] proposed design principles for versatile web query

languages. It provides efficient and effective access to data on the web. Versatile query languages able to query data in any of the heterogeneous representation formats used in both the standard and semantic web query activities.

John D King [7] presents the problem of how to find what information is contained in each search engine, what bias a search engine may have, and how to select the best search engine for a particular information need. To solve all these problems they introduce a new approach called search engine content analysis. A search engine content analysis is a new development of traditional information retrieval field called collection selection, which deals with general information repositories. Current research in collection selection relies on full access to the collection or estimations of the size of the collections and the collection descriptions are represented as term occurrence statistics.

Krishna Bharat [8] presents an extension to search engines called *SearchPad* that makes it possible to keep track of "search context" explicitly and describes an efficient implementation of this idea deployed on four search engines: *AltaVista*, *Excite*, *Google* and *Hotbot*. The design of *SearchPad* has several desirable properties: (i) portability across all major platforms and browsers, (ii) instant start requiring no code download or special actions on the part of the user, (iii) no server side storage, and (iv) no added client-server communication overhead. An added benefit is that it allows search services to collect valuable relevance information about the results shown to the user. In the context of each query *SearchPad* can log the actions taken by the user, and in particular record the links that were considered *relevant* by the user in the context of the query. The service was tested in a multi-platform environment with over 150 users for 4 months and found to be usable and helpful. They discovered that the ability to maintain search context explicitly seems to affect the way people search. Repeat *SearchPad* users looked at more search results than is typical on the web, suggesting that availability of search context may partially compensate for non relevant pages in the ranking.

Adam D Bradley et. al. [9] presents a novel web protocol called as “Basis Token Consistency” (BTC). This protocol allows compliant caches to guarantee strong consistency of content retrieved from supporting servers. Then they compare the performance of BTC with the traditional TTL (Time To Live) algorithm under a range of synthetic workloads in order to illustrate its qualitative performance properties. Ricardo Baeza-Yates et. al. [10] explore the impact of different approaches, such as static vs. dynamic caching, and caching query results vs. caching posting lists. Using a query log spanning a whole year, they explore the limitations of caching and demonstrate that caching posting lists can achieve higher hit rates than caching query answers. Authors propose a new algorithm for static caching of posting lists, which outperforms previous methods and study the problem of finding the optimal way to split the static cache between answers and posting lists. Finally, they measure how the changes in the query log influence the effectiveness of static caching, given observation that the distribution of the queries changes slowly over time. The results and observations are applicable to different levels of the data-access hierarchy, for instance, for a memory/disk layer or a broker/remote server layer.

Evangelos P. Markatos [11] explore the problem of *Caching of Search Engine Query Results* in order to reduce the computing and I/O requirements needed to support the functionality of a search engine of the World-Wide Web. The paper shows that it is possible to cache Search Engine results using moderate amounts of memory and small computational overhead. The contributions of the paper are: They study the traces of a popular Search Engine (Excite) and show that there is a significant locality in the queries asked, that is, 20%-30% of the queries have been previously submitted by the same or a different user. Using trace-driven simulation shows that medium-sized accelerator caches are enough to hold the results of most of the repeatedly submitted queries. They compare caching of the most popular queries (static caching) with caching of the most recently accessed queries (dynamic caching) and show that static caching of query results is promising approach for small caches, while dynamic caching has significantly better performance for large caches.

Dharmendra Patel et. al. [12] introduced one prediction model which predicts sequences of web pages in advance and stores all web pages in cache memory of proxy server when user starts a session and as a result access latency to access web pages can be reduced. This prediction model consists of several components to do correct prediction. The components of prediction models are Pre-processing, User Session Identification, Pattern Generation and Pre-fetching. This paper introduces pre-processing component of prediction model. The algorithm of pre-processing work is described with result and comparison of proposed work is made among Markov model,

Popularity based model and LRS model. The paper concludes that other model clean some useful web pages with unnecessary pages while this proposed algorithm make sure of that thing.

In paper [13], the comparison of Semantic based multiple search engines and standard search engine is evaluated and the analization of both advantages and disadvantages of some current Web cache replacement algorithms including Lowest Relative Value algorithm (LRV), Least Weighted Usage algorithm (LWU) and Least Unified-Value (LUV) algorithm is done. A novel algorithm called Least Grade Replacement (LGR) is proposed, which takes recency, rate of recurrence, ideal-history, and document size into account for Web cache optimization. The use of proxy server reduces the work load for the main server, thus increasing the performance of the servers. The simulation observations showed that the novel algorithm (LGR) is enhanced than LRV, LUV and LWU in terms of hit ratio (HR) and byte hit ratio (BHR).

3. System Architecture

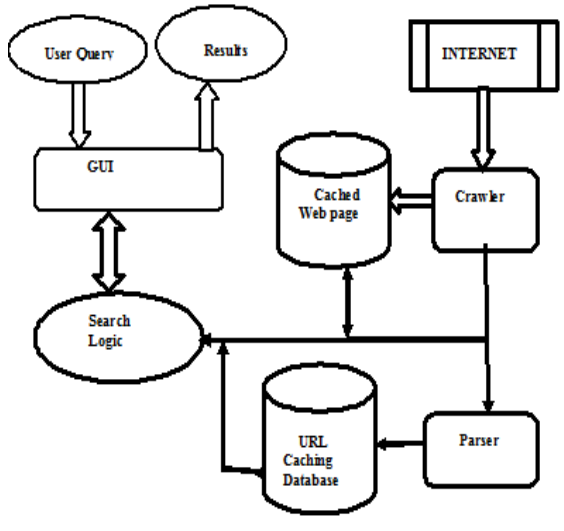


Figure 1: System Architecture of Web Caching for multiple search engines

Architecture contains GUI (Graphical User Interface), search module, web cache, URL caching database, parser, crawler, Internet. User is provided with GUI where user can enter the search keyword and user can also select the Search engines. Search results are displayed back in GUI. Results are given with performance comparison. Search module gets the input from user , then it search for the key word in URL caching database if key word is present it retrieves corresponding URLs from database with search engine information. If search keyword is not present in database then it searches in internet. Cached web pages are stored in local disk and its path is stored in database. A crawler program collects the Web pages on the Internet. The collected Web pages are transported to a Web page database to be stored for the use of future retrieving URLs and corresponding Web pages. Parser is used to parse semantic web pages and normal web pages. Database of both web objects and cached URL keeps only last 15 days content. If it is older than 15 days it is deleted by the caching program.

4. Problem definition

The large numbers of web pages are stored in the database, then we design Cached Search engine to search the resulting web pages links which are exactly matched in the static database and clustered using Hash table Clustering Algorithm (HTCA) [1] and we proposed Cached Search Algorithm (CSA) on top of the multiple search engines like Google, Yahoo and Bing, our objectives are:

- To reduce response time of the search engine.
- To increase the response throughput of searched results.

4.1 Algorithm

Table 1**:** Notations used in algorithms

Symbol	**Meaning**
C_f	Cached File
F	File
W_p	Web page
R_L	Resulting Files List
n	Number of files
m	Number of Keywords
k	keyword
TempList	Temporary List of Files

The Table 1 shows the notations used in algorithms. We have proposed the Cached Search Algorithm to extract the web pages that exactly match with all the keywords from the database or from the web cache and the resulting web pages are clustered using Hash Table Clustering Algorithm [1].

Table 2: **Cached Search Algorithm (CSA)**

Begin

Step 1: Remove the stop words from the query entered by the user.

Step 2: Check whether the keywords are appeared in Cached file C_f .

Step 3: *if* (k is found in C_f)then
corresponding is retrieved and stored into R_L.
else
Search all the files and folders present in the Current path and stored it into List.

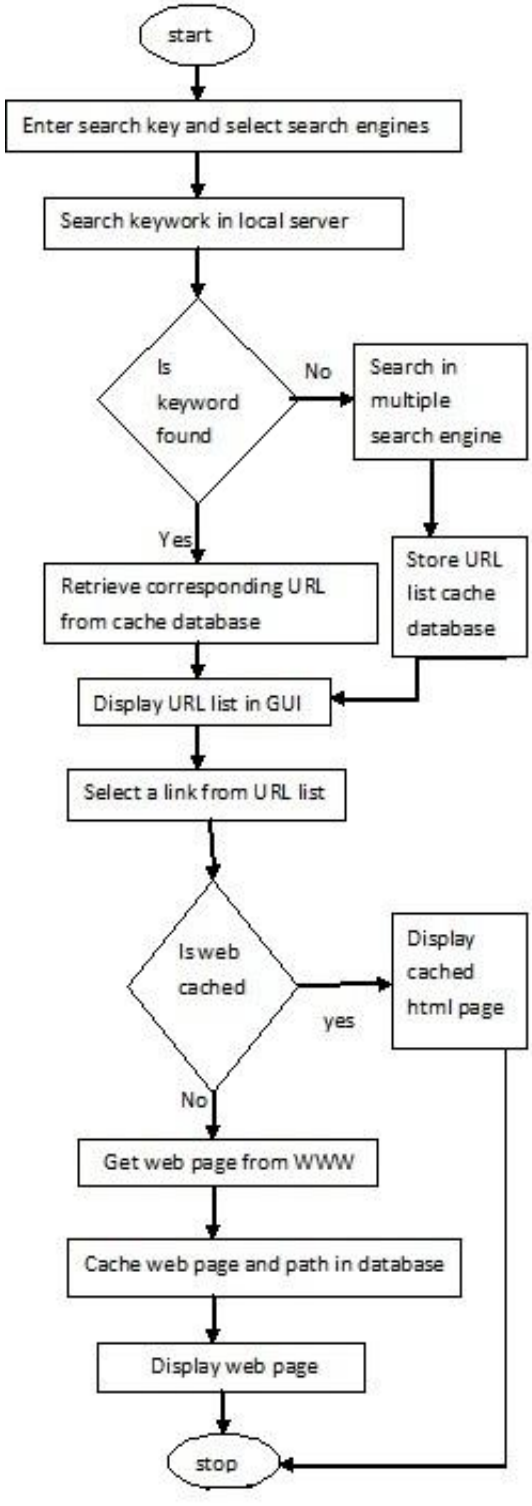
foreach file F *do*
if (F is an.html or .txt file)
TempList = TempList + F
else
Delete the file F from the list
end if
end for
for i = 1 to n *do*
for j = 1 to m *do*
if ($k_j \in F_c$)
$R_L = R_L + F$
Add k and the Pagelink into C_f
else
Discard the File F
end if
end for
end for
end if

Step 4: return R_L

End

5. Implementation

The Flowchart is described in the following steps:



Step 1: Web caching flow initiates when user enters search string and selects multiple search engines to

compare the result and to get the cached or actual web page from WWW.

Step 2: Application checks the keyword in local database.
Step 3: If keyword is found in local database then application fetches corresponding URL list and displays in GUI.

Step 4: If keyword is not found in cache database, then application freshly fetches from multiple search engine and stores in URL database.

Step 5: Select a link from URL list.

Step 6: If URL is present in cached URL database, then time of caching is tested. If it is older than 15 days then URL link is deleted from the database, or else cached web page is displayed.

Step 7: If selected web page is not present in cache, then it is crawled from WWW and cached link is stored in database for future reference.

Step 8: Retrieved web page is displayed in browser.

6. Web services used

In order to get the search results from multiple search engines we have used open source web services of Google, Yahoo and Bing.

Google:

Google provides web services through which developers can fetch the search result from Google that may be web, video, image. When a program request through Google web services, the Google responds back through XML or JSON format. To access Google web services user need to create application id. Google controls the request by app id. It provides 100 query per day per app id, if user wants more number of query then user need to subscribe corresponding package.

Google provides web services through following link
https://ajax.googleapis.com/ajax/services/search/web?v=1.0&q=<Query-String>

Yahoo:

Developers can fetch yahoo search results through web services. To access yahoo web services developer need to create a app id. Once app id is created then developer can request web services to get the result from yahoo. Yahoo responds to the requests by XML format or JSON format.

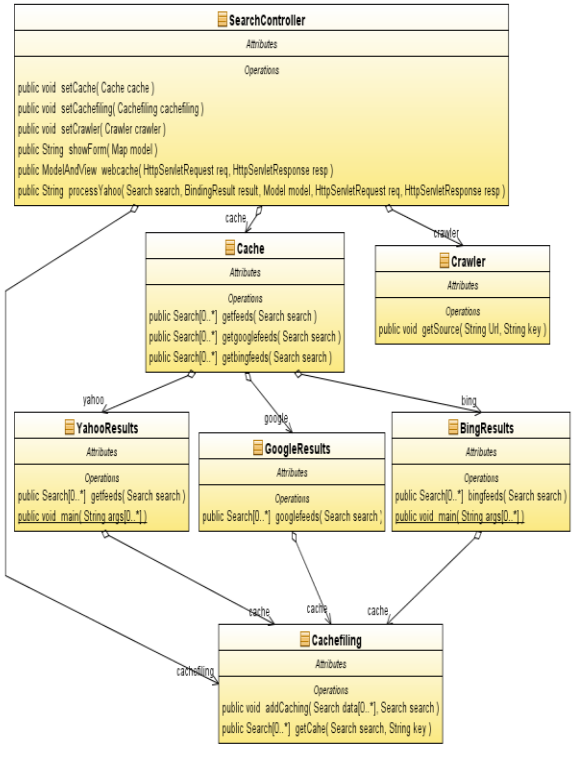
Yahoo web services can be accessed by following URL,
http://api.search.yahoo.com/WebSearchService/V1/webSearch?appid=YahooDemo&query=<Query-String>

Bing:

Developers can fetch yahoo search results through web services. To access yahoo web services developer need to create a app id. Once app id is created then developer can request web services to get the result from yahoo. Yahoo responds to the requests by XML format or JSON format.
Bing link to access web service,
http://api.search.live.net/json.aspx?Appid=8EB910AC13B632EA2F101793EC72C7F8CAA12300&query=<QueryString>&sources=web

Class Diagram:

The Figure 2 shows the Class Diagram of the System Model.



7. Performance Analysis

The performance is analyzed based on WEPS dataset and Mobile dataset of web (static web pages stored in database) and web results given by multiple search engines (i.e. dynamic web pages). Performance is analyzed based on responding time of multiple search engines. We developed the Cached Search Engine using java Programming Language and we tested our search engine by two datasets.

We have used Google speed tracer to find the responding time of web application. Speed Tracer is a tool to help to identify and fix performance problems in web applications. It visualizes metrics that are taken from low level instrumentation points inside of the browser and analyzes them as application runs. Speed Tracer is available as a Chrome extension and works on all platforms where extensions are currently supported (Windows and Linux).

7.1 Test Data

Initially, we downloaded 500 web pages which are relevant to mobile data and store it in a static database and called it as test data.

Figure 3: Snapshot of Cached Search Engine

The Figure 3 shows that snapshot of the Cached Search Engine. A user can Search the mobile by entering the name of the mobile in the text box besides *Enter the keyword* and press the *search* button. Once the keyword has taken then start searching the keyword in the cache database, if it's found then URL of corresponding web page will be displayed in the List box. Otherwise, it takes results from the static database. Here the results are clustered based on HTCA algorithm. When the user click on the URL, the corresponding web page brief description displayed in the Description box, user can go through description to select or not to select the resultant web page.

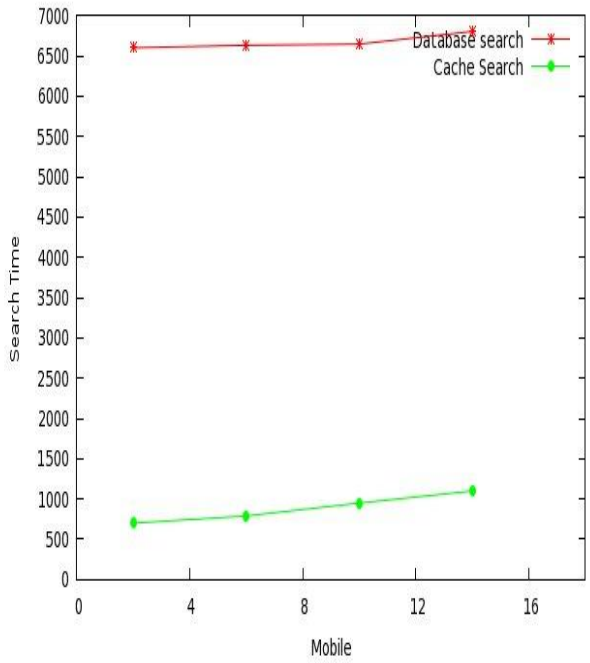


Figure 4: Comparison of response time of with v/s without caching.

7.2 Training data

Here, we considered training data as Web People Search Dataset. In our previous work [1], there is no caching concept for Web People Search Engine. Therefore, we took this dataset as training data for this work. The Figure 5 shows the comparison of the response time for with v/s without caching for WEPS dataset.

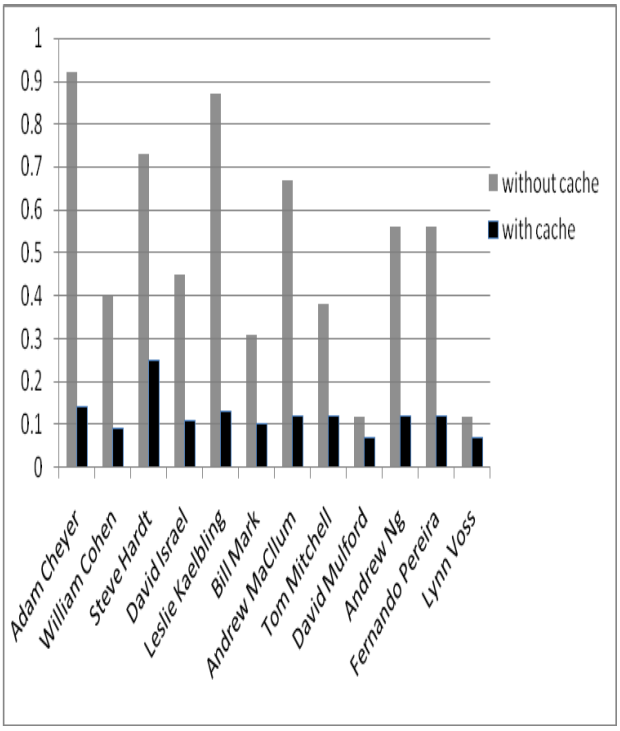


Figure 5: Comparison of the response time for with v/s without caching for WEPS dataset.

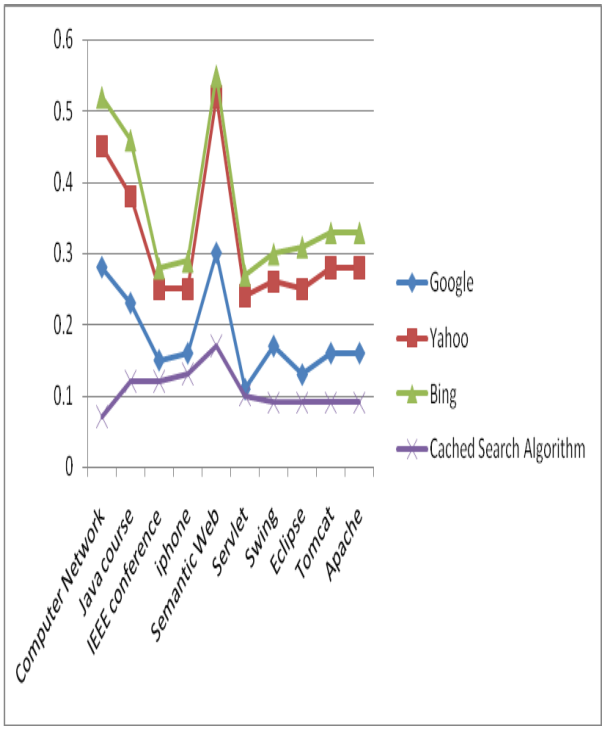


Figure 6: Comparison of the Response time of multiple search engines with the Cached Search Algorithm.

The Figure 6 shows that comparison of the Response time of Cached Search Algorithm on top of the multiple search engines like Google, Yahoo and Bing search engines. We compared the response time of the multiple search engines for 2000 words. The graph is plotted for only 10 words. The performance of the Google search engine is better than Yahoo and Bing search engines. The Cached Search Algorithm outperforms better response time compared to other search engines.

8. Conclusions

In this paper, our contribution can be split into two parts. In the first part, we proposed Cached Search Algorithm (CSA) on top of the multiple search engines like Google, Yahoo and Bing and achieved the better response time while accessing the resulting web pages. In the second part, we design and implemented the Cached Search Engine and the performance evaluated based on the training data (WEPS dataset [1]) and the test data (Mobile dataset). The Cached Search outperforms the better by reducing the response time of search engine and to increase response throughput of the searched results.

9. References

[1] C.N. Pushpa, J. Thriveni, K.R. Venugopal, and L.M. Patnaik, "Enhancement of F-measure for Web People Search using Hashing Technique", *International Journal of Information Processing*, vol. 5, no. 4, IK International Publishing House Pvt. Ltd., India, 2011, pp. 35-44.

[2] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine", *In Proceedings of the Seventh World Wide Web Confernce*, 1998.

[3] Qingqing Gan and Torsten Suel, "Improved Techniques for Result Caching in Web Search Engines", *In Proceedings of International World Wide Web Conference Comittee (IW3C2)*, Madrid, Spain, April 2009, pp. 431-440.

[4] Junghoo Cho, Hector Garcia-Molina, and Lawrence Page, " Efficient crawling through URL ordering", *Journal Computer Networks and ISDN Systems , In Proceedings of the seventh international conference on World Wide Web 7*, Elsevier Science Publishers B. V. Amsterdam, The Netherlands, 1998, pp. 161-172.

[5] Jon M Kleinberg, "Authoritative Sources in a Hyperlinked Environment", *Journal of the ACM*, vol. 46, no. 5, September 1995, pp. 604-632.

[6] François Bry, Christoph Koch, Tim Furche, Sebastian Schaffert, Liviu Badea, and Sacha Berger, "Querying the Web Reconsidered: Design Principles for Versatile Web Query Languages", *Semantic Web-based Information Systems*, 2007.

[7] John D King, "Search Engine Content Analysis", *Thesis*, Queensland University of Technology, Brisbane, Australia, December 2008.
[8] Krishna Bharat, "SearchPad: Explicit Capture of Search Context to Support Web Search", *9th International WWW Conference (WWW9)/ Computer Networks*, vol. 33(1-6), 2000, pp. 493-501.
[9] Adam D Bradley and Azer Bestavros, "Basis on Token Consistency: Supporting Strong Web Cache Consistency", Global Telecommunications Conference(GLOBECOM'02) IEEE, 2002.
[10] Ricardo Baeza-Yates, Aristides Gionis, Flavio P. Junqueira, Vanessa Murdock, and Vassioulis Plachouras, "Design Trade-Offs for Search Engine Caching", *ACM Transactions on the Web*, vol. 2, no, 4, Article 20, October 2008.
[11] Evangelos P. Markatos, "On Caching Search Engine Query Results", *In Proceedings of the 5th International Web Caching and Content Delivery Workshop*, 2000.
[12] Dharmendra Patel, Atul Patel and Kalpesh Parikh, "Preprocessing Algorithm of Prediction Model for Web Caching and Perfecting", *International Journal of Information Technology and Knowledge Management*, July-December 2011, vol. 4, no. 2, pp. 343-345.
[13] S. Latha Shanmuga Vadivu and M. Rajaram, "Optimization of Web Caching and Response Time in Semantic based Multiple Web Search Engine", *European Journal of Scientific Research* ISSN 1450-216X vol. 56 no.2 2011, pp.244-255.

Pushpa C N has completed Bachelor of Engineering in Computer Science and Engineering from Bangalore Universi--ty, Master of Technology in VLSI Design and Embedded Systems from Visvesvaraya Technolog--ical University. She has 12 years of teaching experience. Presently she is working as Assistant Professor in Department of Computer Science and Engineering at UVCE, Bangalore and pursuing her Ph.D in Semantic Web.

Thriveni J has completed Bachelor of Engineering, Masters of Engineering and Doctoral Degree in Computer Science and Engineering. She has 4 years of industrial exp--erience and 16 years of teaching experience. Curren--tly she is an Associate Professor in the Department of Computer Science and Engineering, University Visvesvaraya College of Engineering, Bangalore. Her research interests include Networks, Data Mining and Biometrics.

Venugopal K R is currently the Principal, University Vi--svesvaraya College of Eng- ineering, Bangalore Univer- -sity,Bangalore. He obtained

his Bachelor of Engineering from University Visvesv--araya College of Engg. He received his Masters degree in Computer Science and Automation from Indian Institute of Science Bangalore. He was awarded Ph.D. in Economics from Bangalore University and Ph.D. in Computer Science from Indian Institute of Technology, Madras. He has a distinguished academic career and has degrees in Electronics, Economics, Law, Business Finance, Public Relations, Communications, Industrial Relations, Computer Science and Journalism. He has authored 31 books on Computer Science and Economics, which include Petrodollar and the World Economy, C Aptitude, Mastering C, Microprocessor Programming, Mastering C++ and Digital Circuits and Systems etc.. During his three decades of service at UVCE he has over 250 research papers to his credit. His research interests include Computer Networks, Wireless Sensor Networks, Parallel and Distributed Systems, Digital Signal Processing and Data Mining.

L M Patnaik is a Honorary Professor in Indian Institute of Science, Bangalore. During the past 35

years of his service at the Institute he has over 700 research publications in refereed International Journals and refereed International Conference Proceedings. He is a Fellow of all the four leading Science and Engineering Academies in India; Fellow of the IEEE and the Academy of Science for the Developing World. He has received twenty national and international awards; notable among them is the IEEE Technical Achievement Award for his significant contributions to High Performance Computing and Soft Computing. His areas of research interest have been Parallel and Distributed Computing, Mobile Computing, CAD for VLSI circuits, Soft Computing and Computational Neuroscience.