Volume 2, Issue 3, March 2012 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Improving the Performance of Web Using Enhanced Prefetching Algorithm

K.Ramu[1], Dr.R.Sugumar[2] and B.Shanmugasundaram[3]
[1] Research Scholoar, Department of CSE,
Bharath University,Chennai, Tamilnadu, India

[2] Department of IT, R.M.D.Engineering College
Chennai, Tamilnadu, India

[3]Department of CSE, Gojan School of Business &Tech
Chennai, Tamilnadu, India

ABSRACT-Due to the fast development of internet services and a huge amount of network traffic, it is becoming an essential issue to reduce World Wide Web user-perceived latency. Web prefetching is a technique focused on web latency reduction based on predicting the next future web object to be accessed by the user and prefetching it in idle times. So, if finally the user requests it, the object will be already at the client's cache. The basics of web prefetching techniques preprocess the user requests, before they are actually demanded. Therefore, the time that the user must wait for the requested documents can be reduced by hiding the request latencies. In this work, we introduce a simple and transparent enhanced prefetching algorithm which combines both the top 10 and next-n prefetching approaches. In addition to using access-frequency as the criteria for prefetching, the proposed algorithm also use the time of access of web documents to generate the top 10 list.
Keywords: Web cache, Web prefetch, Latency, Access time

I. INTRODUCTION

Web prefetching is an effective tool for improving the access to the World Wide Web. Prefetching can be initiated either at the client side or at the server side .The benefit of web prefetching is to provide low retrieval latency for users, which can be explained as high hit ratio. Prefetching also increases system resource requirements in order to improve hit ratio. Resources consumed by prefetching include server CPU cycles, server disk I/O's, and network bandwidth. The prefetching technique has two main components: The prediction engine and the prefetching engine. The prediction engine runs a prediction algorithm to predict the next user's request and provide these predictions as hints to the prefetching engine. The prefetching engine handles the hints and decides to prefetch them or not depending on some conditions like available bandwidth or idle time. Wcol introduced by Chinen and Yamaguchi [2004] is a prefetching proxy server for WWW. This program prefetches referred pages (first n embedded images and m linked documents to be more precise) from a user-retrieved page. This scheme suffers from the drawback that it is a deterministic approach and necessarily prefetches a specified number of web objects linked to the requested web page. This could lead to network congestion and could result in prefetching large number of useless objects. The scheme has the advantage of being very simple to implement.

The goal of this article is to design an enhanced prefetching algorithm that could be deployed at the proxy level of network architecture. The focus of the work is to make use of the popularity of the web documents requested by the clients. To improve cache performance, researches have introduced web prefetching to work in conjunction with web caching, which means prefetching web documents from web servers, even before the user requests them. Prefetching techniques rely on predictive approaches to speculatively retrieve and store web objects into the cache for future use. Predictions on what to prefetch are made based on different criteria such as history, popularity and content.

II. WEBPREFETCHING

Web prefetching is a technique for reducing network latencies. This area of research gains impor-

tance since, an user always expects an interactive response, better satisfaction and quality of output. The prefetching can be defined as "A reference that misses in the object cache is prefetchable for a peak period if the object:

- is cacheable
- exists at least since the beginning of the previous off-peak period
- is not modified since the beginning of the previous off-peak period

Now the prefetching can be defined according to World Wide Web as "A Web resource is prefetchable if and only if it is cacheable and its retrieval is safe". A number of commercial systems used today implement some form of prefetching. There are also a number of browser extensions for Netscape and Microsoft Internet Explorer as well as some personal proxies that perform prefetching of links of the current page. Many research papers have been published on the use of prefetching as a mechanism to improve the latencies provided by caching systems. There are various types of Prefetching techniques exist namely, Web Prefetching , Data Prefetching and Other issues, Web Prefetching is further classified into Cache Prefetching , Proxy Prefetching and Semantic Prefetching , Data Prefetching is classified into Content Prefetching and Context Prefetching.

III. DEPLOYMENT OF PREFETCHING ALGORITHMS

Three main locations in the Internet where prefetching algorithms can be deployed are:

A. Between browser clients and web servers.

In this technique, the server computes the likelihood that a particular web page will be accessed next and conveys the information to the clients. The client program then decides whether of not to prefetch the page. The prediction is done by a prediction by Prediction by Partial Match (PPM) modal. A dependency graph is constructed that depicts the pattern of accesses to different files stored at the server. The graph has a node for every file that has ever been accessed.

B. Between proxies and web servers.

In this technique, the Web servers push the pages to the proxies regularly. Without prefetching the performance of the proxies is limited. This scheme further reduces the latency. Geographical Push-Caching where a Web server a Web server selectively sends it documents to the caches that are closest to its clients. Here the important issues is to maintain the accurate network topology.

C. Between browser clients and proxies.

The web prefetching can also be done between browser clients and proxies. One approach is to predict which cached documents a user might reference next (based on PPM) and take the advantage of idle time between user requests to push the documents to the users. The first two approaches run the risk of increasing wide area network traffic, while the last one only affects the traffic over the modems or the LANs. All of these approaches attempt to prefetch either documents that are considered as popular at servers or documents that are predicted to be accessed by user in the near future based on the access pattern.

IV. REQUIREMENTS OF WEB PREFETCHING ALGORITHMS

The web prefetching algorithms should be carefully designed otherwise it can compromise the overall efficiency of the network and could have adverse effects on the network architecture. There are three major requirements for a good web prefetching algorithms namely minimize the user access latency, maximize the prefetch hit ratio and maximize the cache hit ratio.

A. Minimize the User Access Latency

Delays in access to Web based Information continues to be a serious problem even with higher network bandwidth, due to overhead web latency has increased due to which web performance has decreased. User perceived latency from several sources such as bandwidth, speed, overhead, accessing the web page etc. In accordance of "Eight Second Rule", it can be observed that web latency affects the user work and lot of efforts is taken to minimize the latency perceived by the user. Caching of web documents has been developed to reduce the latency but it has the drawback that it stores the pages without any prior knowledge i.e. the hit ratio is less. Web prefetching is an effective technique to minimize user's web access latency.

B. Maximize the Prefetch Hit Ratio and Cache Hit Ratio

The Hit Ratio is a ratio of the requests that are serviced from prefetched cache to the total number of requests. Higher Hit Ratio can contribute more to the improvement of the efficiency of cache. One way to further increase the cache hit ratio is to anticipate future requests and prefetch these objects into a local cache. On the other hand, prefetching consumes more network bandwidth. Web prefetching has been recognized as an effective solution to maximize the hit ratio.

V. DESIGN OF ENHANCED PREFETCHING ALGORITHM

The major objective of this paer is to design and implement an Enhanced Prefetching Algorithm

(EPA) which combines both the top 10 and next-n prefetching approaches. In addition to using access-frequency as the criteria for prefetching, the proposed system also use the time of access of web documents to generate the top 10 list. This approach of using access-frequency and time of access is known as the Greedy Dual Size (GDS) Policy approach, which has been used in cache management. Instead of generating next-n list for all the documents accessed by the users, the system log the next-n documents for the top 10 documents only, thus reducing complexity and overhead. The Enhanced Prefetching Algorithm is shown in Table I.

The idea of the EPA algorithm is to keep the top ten popular documents for each web server, by this means, clients or proxy servers can prefetch only these popular documents without significantly increasing network traffic. The proposed result shows that this approach expects more than 30% of client requests and achieves close to a 65% hit ratio at the cost of increasing network traffic b y - no more than 15% in most cases. This experiment is close to the prefetching by Popularity algorithm. The algorithm keeps copies of n most popular objects in the cache and updates them immediately whenever these objects are modified. From the Zipf-like distribution, we know that popular objects are responsible for majority of requests from users. If we keep in our cache copies of popular objects which are most likely to be requested, this will definitely achieve the highest possible hit ratio. On the other hand, its bandwidth consumption is high.

Table I Enhanced Prefetching Algorithm

```
Procedure Prefetch (Array R, int M, float maxSize)
//h, b are sequences of document ids
Begin
    1. PrefetchSeq = Ø
    2. for each rule h -> b such that h < R
    3. for each dUb such that d.size < maxSize
    4. PrefetchSeq = prefetchSeq U d
    5. end for
    6. end for
    7. Sort documents in prefetchSeq in decreasing order of the confidence of the corresponding rule and keep the first M ones.
    8.
        Reurn
        prfetc
        chSeq
End
```

The proposed EPA algorithm uses as input such global access statistics as (1) estimates of object reference frequencies and (2) estimates of object lifetimes. This estimation can be well maintained by content distribution servers if they can collaborate between each other. Content serverscollect user access statistics and publish information of objects popularity and patterns of usage; gather usage reports from their users, aggregate and analyze them and make them available to each other. They can also send prefetching hints to each other or actively push to each other objects that are likely to request in the near future. Whenever the replicated object is updated in the original server, the new version of it will be sent immediately to any cache that has subscribed.

The Architecture of Enhanced Prefetching Algorithm is illustrated in fig.1.In the proposed architecture, 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. 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 is analyzed in, where the computations required for query executions are optimized

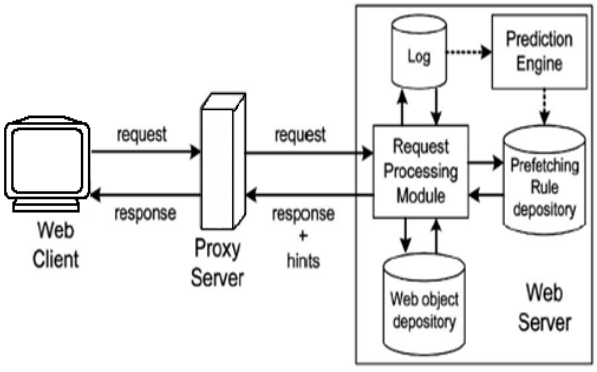

Fig.1 Architecture of Enhanced Prefetching Algorithm

VI. SIMULATION AND RESULTS

The proposed system has performed all experiments on a PC with Pentium IV and 2 GB RAM, under the Windows XP using VB.Net. The performance of the proposed Enhanced prefetching algorithm is compared with the existing GDS for long-term Web prefetching. GDS prefetching is based on document size and has an elegant aging mechanism.

Similarly, the Greedy-Dual-Frequency (GDF) policy takes into account file frequency and exploits the aging mechanism to deal with cache pollution. The efficiency of a cache replacement policy can be evaluated along two popular metrics: file hit ratio and byte hit ratio. Using four different web server logs, it shows that GDS-like replacement policies emphasizing size yield the best file hit ratio but typically show poor byte hit ratio, while GDF-like replacement policies emphasizing frequency have better byte hit ratio but result in worse file hit ratio. The Greedy-Dual-Frequency-Size policy which allows to balance the emphasis on size and frequency.

The objective of the proposed EPA algorithm is to make use of the popularity of the web documents requested by the clients. The popularity of web documents was calculated using two parameters the time difference between two consecutive requests for the same document and the access frequency of the same document.

The following parameters are used to calculate performance metrics of the EPA algorithm:

- **Cache Hit**: A cache hit is assessed when a user requested document is fetched from cache.
- **Cache Miss**: A cache miss is assessed when a user requested document is not present in cache.
- **Top-10 Hit**: A top-10 hit is assessed when a user requested a document that is present in top-10 cache.
- **Total Requests**: This is the total number of requests made by clients during the testing phase.

The metrics used to demonstrate the efficiency of the EPA algorithm are:

- **Cache Hit Ratio**: If requested data is contained in the cache (cache hit), this request can be served by simply reading the cache. This is the ratio of the total number of cache hits to the total number of requests during the testing phase.
- **Prefetch Accuracy**: Prefetch accuracy is the percent of prefetches that are accessed by demand fetches before they are evicted from the cache.
- **Prefetch Hit Ratio**: This is the ratio of the total number of prefetch hits to the total number of requests where a prefetch hit is the sum of top-10 hits and next-n prefetch hits. This measures the usefulness of predictions.

A. Improving the Prefetch Hit Ratio Using EPA Algorithm

This research work concentrates on the various performance metrics such as prefetch accuracy and prefetch hit ratio. The first metric used to describe the performance of EPA algorithm is prefetch hit ratio. The experiments are performed on different cache size of 256 MB, 512 MB and 1024 MB. Fig.2 illustrates the graph of Prefetch Hit ratio and Cache size for GDS and EPA algorithm. It can be seen that hit ratio of the proposed EPA algorithm has improved compared to the GDS algorithm. Whenever the hit ratio of a prefetching algorithm is increased, then the specific document will be moved into top-10 cache.

Table II Comparison of Hit Ratio and Cache size for GDS and EPA algorithm

Cache Size (MB)	PrefetchHit Ratio of GDS algorithm (%)	Hit Ratio of EPA algorithm (%)	% of Hit Ratio improved
256	31.52	52.94	41.63
512	47.37	69.53	48.57
1024	76.51	91.12	62.42

From the Table II, it is clear that the prefetchhit ratio of the proposed algorithm is improved to 41.63%, 48.57%, and 62.42 for different cache size 256 MB, 512 MB RAM and 1024 MB RAM respectively.

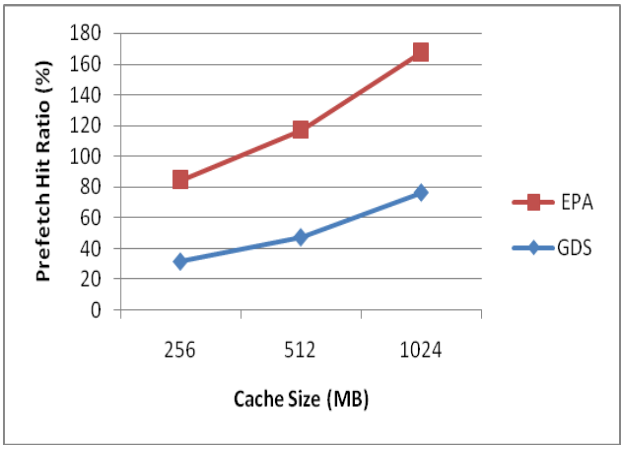

Fig.2 Performance Comparison of Prefetch Hit Ratio vs Cache size

B. Improving the Prefetch Accuracy Using EPA Algorithm

The next metric which is used to demonstrate the performance of web prefetching algorithm is Prefetch Accuracy. The experiments are performed on different cache size of 256 MB, 512 MB and 1024 MB. The Fig.3 illustrates a graph of Prefetch Accuracy and Cache Size for GDS and EPA algorithm. It can be seen that hit ratio of the proposed EPA algorithm has improved to 42.78%, 50.09%, and 63.42 for different cache size of 256 MB, 512 MB and 1024 MB compared to the GDS algorithm. Whenever the prefetch

accuracy of a prefetching algorithm is increased, then the specific document will be moved into top-10 cache.

TABLE III Comparison of Accuracy and Cache size for GDS and EPA algorithm

Cache Size (MB)	Prefetch Accuracy of GDS algorithm (%)	Prefetch Accuracy of EPA algorithm (%)	% of Accuracy improved
256	35.19	56.35	42.78
512	45.37	64.53	50.09
1024	78.24	94.62	63.42

From the Table III, it is clear that the accuracy of the proposed algorithm is improved to 42.78%, 50.09%, 63.42 for different cache size 256 MB RAM, 512 MB RAM and 1024 MB RAM respectively.

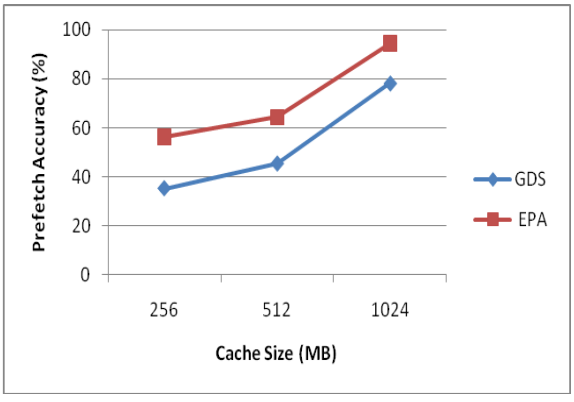

Fig.3 Performance Comparison of Accuracy vs Cache size

C. Improving the Cache Hit Ratio Using EPA Algorithm

The next metric which is used to demonstrate the performance of web prefetching algorithm is cache hit ratio. The experiments are performed on different cache size of 256 MB, 512MB and 1024 MB. The Fig.4 illustrates a graph of cache hit ratio and Cache Size for GDS and EPA algorithm. It can be seen that hit ratio of the proposed EPA algorithm has improved to 44.37%, 49.70%, 53.32for different cache size of 256 MB, 512MB and 1024 MB compared to the GDS algorithm. Whenever the cache hit ratioof a prefetching algorithm is increased, then the specific document will be moved into top-10 cache.

TABLE IV Comparison of Cache hit ratio and Cache size for GDS and EPA algorithm

Cache Size (MB)	Cache Hit Ratio of GDS algorithm (%)	Cache Hit Ratio of EPA algorithm (%)	% of Hit Ratio improved
256	38.86	56.27	44.37
512	59.30	81.19	49.70
1024	82.68	90.64	53.32

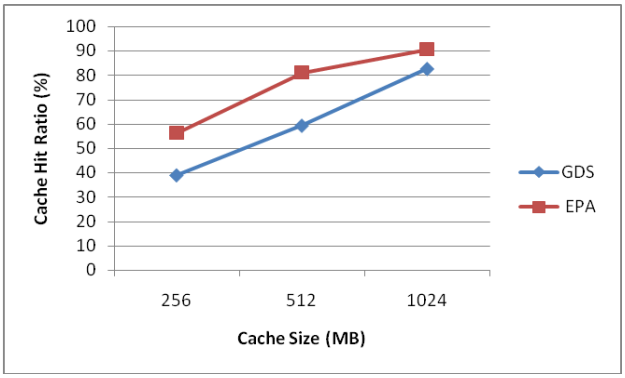

Fig.4 Performance Comparison of Cache hit ratio vs Cache size

From the Table IV, it is clear that the cache hit ratio of the proposed algorithm is improved to 44.37%, 49.70%, and 53.32% for different cache size 256 MB, 512 MB and 1024 MB RAM respectively.

VII. OTHER RESEARCH DIRECTIONS

Web Prefetching is an elegant technique to handle information provisioning for users under these circumstances. In this paper, new strategies for prefetching in a location-aware surrounding are investigated. The mechanism, as well as the underlying location model, could however also be transferred to other application areas such as railway networks and highways. The dynamic service priority allocation and the development of location-aware cache invalidation algorithms are the focus of research in this area. To overcome the problem of long retrieval latency caused by the unpredictable user behaviors during multimedia presentation, a prefetching scheme using the association rules from the data mining technique was proposed. The data mining technique can provide some priority information such as the support, confidence, and association rules which can be utilized for prefetching continuous media. The prefetching policy can predict user behaviors and evaluate segments that may be accessed in near future. The results show that the

prefetching approach can get better latency reduction, even for small cache size.

Web Prefetching can be applied in various domains to improve the system performance. A location-aware prefetching mechanism is introduced that is independent of any additional infrastructure and that gathers information solely over deployed, lowbandwidth wireless links. Location-awareness becomes more and more important for delivering relevant data. The latter is a challenging task when dealing with unstable and low-bandwidth wireless networks, especially in areas that are not or only poorly covered.

VIII CONCLUSIONS

An Enhanced Prefetching Algorithm has been designed and implemented that could be deployed at the proxy level of network architecture. The performance of EPA approach for 256 MB, 512 MB and 1024 MB cache size compared with the existing GDS algorithm. The proposed algorithm used the time of access of web documents to generate the top 10 list. The results obtained from simulations, in terms of cache hit ratio, prefetch hit ratio and prefetch accuracy shows the efficiency of the proposed algorithm as compared to other approaches. This research work has focused on making use of the popularity of the web documents requested by the clients.

REFERENCES

[1] Bin Wu; Ajay D. Kshemakalyani. "Objective-Greedy Algorithms for Long-Term Web Prefetching", Proceedings IEEE International Symposium, NCA'04, U. S. A, 2004.

[2] Bin Wu; Ajay D. Kshemakalyani. "Objective optimal algorithm for long term web prefetching", Journal of IEEE Transactions on Computers, Vol. 55, Issue 1, 2006.

[3] L.Cherkasova, "Improving WWW proxies performance with Greedy-Dual-Size-Frequency Caching Policy", in HP Technical report, 2008.

[4] V. N. Padmanabhan; J. C. Mogul. "Using Predictive Prefetching to Improve World Wide Web Latency", Proceedings ACM Conference, SIGCOMM, 2008.

[5] Evangelos P. Markatos & Catherine E. Chronaki, "A Top-10 Approach to Prefetching on the Web, Proceedings of INET' 98, 1998.

[6] Domenech J., Sahuquillo J., Gil J. A., and A. Pont , "The impact of the web prefetching architecture on the limits of reducing user's perceived latency", Proceedings of /ACM International Conference on Web Intelligence,pp.155-178, 2006.

[7] De la Ossa, J. Sahuquillo, A. Pont, J. A. Gil , "An Empirical Study on Maximum Latency Saving in Web Prefetching", IEEE International Joint Conference on Web Intelligence and Intelligent Agent Technology, vol. 1, pp.556-559,2009

[8] Sarina sulaiman siti, Ajith Abraham shahida Sulaiman, "Web caching and prefetching what, why, and how", Proceedings of IEEE International Symposium on Information Technology,pp.1-8, 2008

[9] Umapathi C. and J. raj, "Prefetching algorithm for improving web cache performance", Journal of application science Asian network scientific information, pp. 3122- 4127, 2008.

[10] Markatos E.P. and Chronaki C.E., "A Top-10 Approach to Prefetching on the Web", Proceedings of 12th INET International Conference, pp.228-241, 2009.

[11] Chen X. and Zhang X., "A popularity-based prediction model for web prefetching", IEEE Computer Society Press Los Alamitos, pp.109-123, 2010.

[12] Chen X, Zhang X., "Popularity-based PPM: an effective web prefetching technique for high accuracy and low storage", Proceedings of the international conference on parallel processing. pp.75-93, 2010.

[13] Patil J. B. and Pawar B. V., "GDSF : A Better Algorithm that Optimizes Both Hit Rate and Byte Hit Rate in Internet Web Servers", International Journal of Computer Science and Applications, Vol. 5, No. 4, pp. 1-10, 2007.

[14] Markatos E. and Chironaki C., "A Top 10 Approach for Prefetching the Web", Proceedings of INET Conference, Geneva, Switzerland,pp.237-249, 2006.

[15] Jin S. and Bestavros A., " Popularity-aware greedydual-size web proxy caching algorithms", Proceedings of the 20th International Conference on Distributed Computing Systems, pp.357-378, 2009.

[16] Zhijie B., Zhimin G. and Yu J, "A Survey of Web Prefetching", Journal of computer research and development, Vol. 46(2), pp. 202-210, 2010.

[17] Jiang Y., Wu M.Y, and Shu W. (2010), "Web prefetching : Costs , benefits and performance", Proceedings of the 11th International World Wide Web Conference, New York,pp.472-483.

[18] Bin W., and Kshemkalyani A. D. (2008), "Objective-greedy algorithms for long-term Web prefetching", Proceedings of Third IEEE International Symposium on Network Computing and Applications, pp.521-542.

Biography

K.Ramu has received his Undergraduate Degree in Computer Science and Engineering from Madras University, in 2002 and the Post Graduate degree in Computer Science and Engineering from Sathyabama Uni-

versity, Chennai in 2005. He is pursuing his PhD in Faculty of Computer Science Engineering from Bharath University, Chennai. He has more than 5 publications in National Conferences and international journal proceedings. He has more than 10 years of teaching experience. His areas of interest include Data Mining, Data Structures, Database Management Systems, Distributed systems and Operating systems.

Sugumar.R has received his Undergraduate Degree in Computer Science and Engineering from Madras University, in 2003 and the Post Graduate degree in Computer Science and Engineering from Dr.M.G.R. Educational and Research Intituite, Chennai in 2007. He has completed his PhD in Faculty of Computer Science Engineering from Bharath University, Chennai in 2011.He has more than 15 publications in National Conferences and international journal proceedings. He has more than 8 years of teaching experience. His areas of interest include Data Mining, Data Structures, Database Management Systems, Distributed systems and Operating systems. He is currently working as an Assistant Professor in the Department of Information Technology at R.M.D.Engineering College, Chennai, India. He is an active researcher in data mining, Web mining and Information Retrieval.

B.Shanmugasundaram has completed his M.C.A from Madurai Kamaraj University during 2002 and completed his M.Tech in Computer Science and Engineering from Dr.M.G.R. Educational and Research Intituite, Chennai during 2011. He has more than 4 years of teaching experience. He has 6 publications in National Conferences and international journal proceedings. His areas of interest include Database Management Systems, Distributed systems OOAD, Software Engineering, Software project management and Computer networks. He is currently working as an Assistant Professor in the Department of Computer Science and Engineering at Gojan School of Business and Technology, Chennai, India.

