Volume 3, Issue 6, June 2013 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Enhanced Dynamic Web Caching: For Scalability & Metadata Management

Deepak Bagga*, Surender Singh
Computer Science & Engineering
Shivalik Institute of Engg .& Technology, Aliyaspur, India

Shilpi Harnal
Department of Computer Science & Application,
Kurukshetra University,
India

***Abstract:** These days web caching suffers from many problems like scalability, robustness, metadata management etc. These problems degrade the performance of the network and can also create frustrating situations for the clients. This paper discusses several web caching schemes such as Distributed Web Caching (DWC), Distributed Web Caching with Clustering (DWCC), Robust Distributed Web Caching (RDWC), Distributed Web Caching for Robustness, Low latency & Disconnection Handling (DWCRLD). Clustering improves the retrieval latency and also helps to provide load balancing in distributed environment. But this cannot ensure the scalability issues, easy handling of frequent disconnections of proxy servers and metadata management issues in the network. This paper presents a strategy that enhances the clustering scheme to provide scalability even if size of the cluster grows, easy handling of frequent disconnections of proxy servers and a structure for proper management of cluster's metadata. Then a comparative table is given that shows its comparison with these schemes.*

***Keywords:** Distributed Web Caching, Clustering, Latency, Robustness, Scalability, Disconnection Handling, Proxy server, clients, metadata*

I. INTRODUCTION

Web has grown rapidly from simple static information and images sharing system to sharing of highly interactive and dynamic services. Now Web has become the most successful application in the internet. It provides quick and easy access to wide number of services and information. As the web is growing exponentially, availability of resources and services is becoming a bottleneck for servers. Many times users have to face frustrating delays while accessing a web page, congestion at servers and frequent disconnections of servers. To maintain its functionality all these latencies must be maintained within the tolerable limits. That's why upgrades in the networks/servers are always required for providing high speed and continuous services to its users. One solution is to store multiple copies of same document but this will increase the storage and maintenance cost. Another solution is to cache only the frequently accessed documents as most of the documents are rather static. This reduces retrieval latency and network traffic as well.

Mosaic [1], were the first web browsers that were capable of caching the web objects for later references. This results into low latency and lower bandwidth consumption. After this start, web caching is rapidly extended to shared caches that can serve multiple clients from a certain organization [2]. By this Hit rates for accessing cached documents at organizational level were enhanced from 30% to 50%. Client-Server architectures are generally suited for these kinds of networks [3].

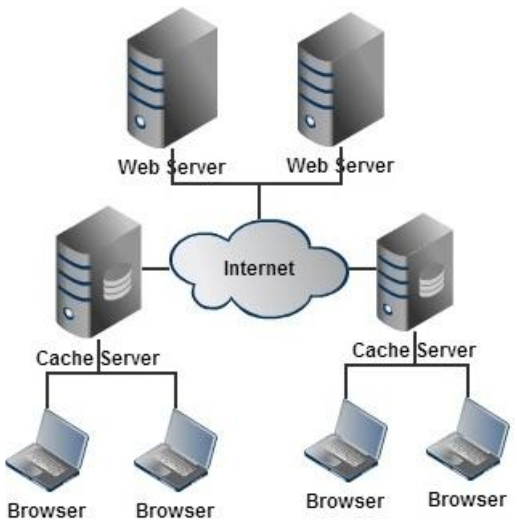

Figure 1: General web caching architecture

Before caching, for every request the documents were fetched directly from the remote located web servers, thus creating a bottleneck for the servers. Further these architectures were enhanced with caching of highly frequent pages at proxy servers to reduce the retrieval delays at servers. As the result of further enhancements, traditional techniques like Hierarchical and Distributed web caching [4], [5], [6] were proposed. Their general architecture consists of proxy servers at some intermediate levels and the main origin server at the top level. In this whenever client made request for a particular document for the first time the request goes directly to the origin server. After receiving documents, client stores them in local cache for future references as cache is capable of storing persistent data. Every time if the requested document is found in the cache, is counted as a Hit otherwise a Miss.

Some good caching schemes are being surveyed by V. Cardellini et al. in [6]. In multilevel hierarchical web caching architecture for every miss the requests were transmitted directly to the upper level proxies and at last to the origin server. This introduces extra levels of delays. This happen because proxy servers at each level of hierarchy were not having information of their neighboring servers at the same level. Thus the concept of metadata exchange between the neighboring proxy servers was introduced. Metadata of proxy servers is the list of pages currently present in their caches. Servers keep exchanging their metadata for fulfilling the requests from clients. This scheme is suitable for environment where metadata is not too large in size. But as the metadata grows in size, metadata exchange itself becomes too much time consuming. Frequent exchange of large sized metadata at each level of hierarchy also increases the network traffic and can cause congestion. This situation can become highly unmanageable.

The solution for this comes in the form of distributed strategy of caching. In this there are no intermediate caches. Only one level of proxy servers are used at the edge of the network. They cooperate with each other to serve the misses. For this Inter Cache Protocol (ICP) is used [7]. By this exchange of metadata becomes manageable to some extent. These caches are also called as cooperating caches. For cooperating caches a hash function [8] can also be used to map client's request for a certain document. But this scheme is limited to local environments only. Also if the number of users and number of registered pages increases the metadata grows in size and results into larger delays while searching into the lengthy metadata. So the scheme of managing larger metadata comes with the concept of clustering of highly correlated proxy servers. This paper is an extension of work done by Rajeev et al. in [9] and in [3]. They have provided a solution for frequent disconnection of servers, low latency and robustness in their scheme with the help of clustering. In our paper we propose a new strategy that is enhancement of their work for achieving scalability and low latency time in clusters for distributed environment. This scheme can also help to make easy and more specific handling of client's request and also provides a manageable structure for metadata handling. Also it provides easy handling for frequent disconnections of proxy servers in the network. The rest of the paper is organized as follows: in the next section we are discussing some of the common problems and issues in web caching. In section 3 we are reviewing some previous work. In section 4 we introduced our proposed strategy along with its working and possible phases. In the next section we are discussing our scheme in comparison with other introduced schemes on the basis of certain evaluation parameters.

II. SOME PROBLEMS AND ISSUES

In this section some general problems and issues related with distributed web caching architecture are discussed.

A. *Extra Overhead:*

Overhead at proxy server increases when it maintains records of all other proxy servers. It also results into more congestion at proxy servers. Each server must keep checking the validity of their data and this leads to extra overhead on servers. Updating and exchanging of metadata on proxy servers keeps them unnecessarily busy.

B. *Size of Cache:*

If size of cache grows it results into larger metadata that becomes unmanageable at proxy servers as they also keeps metadata of all other proxy servers too. In this way cache's size become bottle neck for the maintenance of larger metadata.

C. *Cache Coherence Problem:*

Client must always receive an up-to date data from the proxy server when ever requested. This requirement can result into cache coherence problem. All changes of data at the proxy server cache must be simultaneously reflected to the main server too. If this is not possible there should be some provision to update the cache data for maintaining their coherency.

D. *Scalability:*

As numbers of clients are growing day by day and also clients of one region can request for connectivity, scalability can be an issue as every server has an upper limit to support the client's connections. If the servers provide the connectivity to more clients then they will become congested and will not be able to provide further connectivity. So scalability should be as high as possible. Clustering can be solution for scalability problem. Through this more number of clients can be served and client's data is maintained on cluster basis.

E. *Robustness:*

As proxy servers can serve a limited number of requests, they hang up if the request exceed their limit and started get down the links. In these cases the client request are not fulfill for connections as they always requests to the same proxy server they are connected. These links need to be reset. Clustering can be a solution to this problem and can help to serve most to requests.

F. *Hit Ratio:*

If the requested document is present in the proxy cache, that is called a hit. The hit ratio in web caching should be high enough so that user's request can be served from the pages cached on the cache server instead of forwarding the request to other proxy server. In case of congestion hit ratio will decrease drastically as all the users wait for the requested documents. This degrades the network performance so an approach that can ensures high hit ratio is always an issue.

G. Balancing Of Load:

The scheme of balancing the loads on the proxy servers is a major issue in web caching. If there is no predefined criteria is set the clients can connect to same proxy servers, if there is no limit on the number of clients to proxy servers. This means one server will be overloaded and other proxy server will remain ideal, the busy proxy server get congested and may be down later, so a proper load balancing strategies for proxy server is a major requirement.

H. Low Latency:

As caching provides easy handling of clients request by the proxy servers and speedup the reply process for the clients, so techniques should provide lower delays for all requests as possible.

I. Frequent Disconnections:

Interrupts to continue service can arise due to some unmanageable situations that lead proxy servers to be disconnected. So scheme must be provided that can observe smaller disconnection of proxy server and should be able to maintain consistency and recover back all the Metadata.

J. Staleness:

The pages that are in cache from a longer time are needed to be checked before providing their access [10]. The user has to wait for an expiration check to be finished. No proper mechanisms are present those can provide guarantee of its freshness.

K. Problem with Cookies and Encrypted Files:

Cookies and encrypted pages are always difficult to cache [11]. Limit the use of cookies for dynamic pages only. Even encrypted pages are difficult to store at shared caches.

L. Deletion Hazard:

Sometimes one has deleted own created web pages from the file server if it they are of no future use. But still they remained cached somewhere in any of the proxy server caches. That page might be displayed on trying to access it from different location. Also if one has made changes to a web page and uploaded its new version, proxy at some other site may keep on showing its older version [12].

III. PREVIOUS WORK

A number of caching schemes already exists. An analysis of web caching architectures on the basis of load, connection time latencies and transmission time latencies is done by Pablo Rodrigue, Christian Spanner and Ernst in [2]. Some effective web caching architectures are Single and Multiple caches, Hierarchical Caching, Distributed Web Caching, Cooperating Web Caching Scheme, Distributed Web Caching with Clustering, Hybrid Web Caching and Robust Web Caching Schemes [9].

A. Hierarchical web caching scheme:

A. Chankhunthod et al. in [4] proposed Hierarchical web caching scheme in the Harvest Project, that shares the interest of a large number of clients and also several countries have implemented this scheme. In this architecture caches are placed at the different levels of hierarchy and client's caches are at the bottom level [2]. For every miss the request is redirected to the next upper level caches in the hierarchy. If the document is not present in any of the level, request is forwarded to the origin server and on reply path copy of document is maintained at each intermediate level proxy server. But this scheme incurred many problems such as redundancy of data at each level and longer queries delay. Gadde et al. in [13] presented a Central Directory Approach (CRISP) to be deployed at any level of global or regional cache hierarchy in which certain numbers of caches ties together through a central mapping service to enhance capacity and scalability.

B. Distributed web caching scheme:

Internet Caching Protocol (ICP) [7] and Hyper Text Caching Protocol (HTCP) [14] were designed for distributed environment. They support management, discovery and retrieval of updated data from parent and neighboring caches as well. Cache Array Routing Protocol (CARP) is another approach for distributed caching [8]. In this URL space is divided among an array whose elements are loosely coupled caches. Each document is hashed to a particular cache. Tewari et al. [15] proposes a scheme for fully distributed internet cache architecture, in which location hints are maintained and replicated at all local institutional caches. In Cache Digest [16] and in the Relais Project [17] all caches maintain local directories of contents of other caches for ease of locating documents in other caches. Also caches keep exchanging messages with each other indicating their contents.

C. Hybrid web caching scheme:

In [7] they have presented hybrid scheme, with the concept of caches cooperation at each level in hierarchy. Rabinovich et al. [18] have altered this scheme by limiting the cooperation between the neighboring caches. This scheme advantages by avoiding fetching of documents from the slower or distant caches if that document could be retrieved at a lower cost directly from the origin server.

D. Distributed Web Caching scheme with clustering:

Distributed Web Caching (DWC) scheme supports exchange of metadata among proxy servers periodically to maintain a complete metadata. Thus every proxy server maintains metadata of all other proxy servers along with its own metadata as shown in Figure 2. For every request proxy server firstly checks its own metadata for requested page id, if there is a hit the page is transmitted to the client otherwise it checks for page id in the whole metadata of other proxy servers and if found the request is forwarded to that server else to the origin server. Advantage of this scheme is high hit ratio. But if the size of metadata and number of proxy servers increases it becomes highly unmanageable. Even this scheme fails if any of the proxy servers gets disconnected.

Pid1	Pid2	Pid3	. . . . . .	Pidn

(Pid1,PS1)	(Pid2,PS1)	(Pid1,PS2)	(Pidn,PS2)	. . .			

Figure 2: Metadata maintained at each proxy server

The Distributed Web Caching with clustering (DWCC) [9] and Distributed Web Caching for Robustness, Low latency & Disconnection Handling (DWCRLD) [3] are based on the geographical region based clustering. The author in [9] provides a solution for robustness and scalability problem in web caching due to heavy load. They have used the concept of clustering along with the feature of dynamic allocation of requests by maintaining metadata of neighboring clusters only not of all clusters. They also provide the concept of managing the load of overloaded server by transferring requests to less loaded proxy servers. In [3] the author has refined their scheme of [9] to handle more delays and frequent disconnections of proxy servers. This can result in fastest response to the clients and also provide load balancing. Even these schemes suffer from the scalability problem. If size of the cluster grows, size of metadata grows as well then metadata at every proxy servers can become unmanageable. This problem can be overcome by our enhanced proposed architecture.

IV. THE PROPOSED STRATEGY

This strategy is based upon the Distributed Web Caching with geographical region based Clustering (DWCC). All the proxy servers those are geographically together grouped into the same cluster. This scheme is dynamic in nature and based upon the clustering of proxy servers in distributed environment. This provides easy management of metadata as in previous strategies metadata handing is rather difficult. It will also provide low latency benefit by a factor of s/m as previous strategies has to wait for the metadata exchange before replying for any request. Also load balancing is managed in this by limiting the number of client's requests per cluster. If there is no limit on number of clients per proxy server, some of the proxy servers will be overloaded and starts dropping the requests or will add unlimited delays. This can be handled by limiting the number of clients per proxy server and per cluster [3] as well. This is done by maintaining a cluster queue length data structure.

The proposed strategy includes clients, clusters of proxy servers and origin servers as shown in Figure 3. Every cluster has an extra node than proxy servers that is Metadata Server (MDS). This Metadata server is actually a kind of data base server whose task is to maintain metadata of that cluster and they also keep track of neighboring cluster's metadata. All other proxy servers of a cluster are connected with their Metadata server. Every proxy server only maintains metadata of its own only and does not bother about metadata of other proxy servers of same cluster or of neighboring clusters. In previous strategy [3] every proxy server itself maintains metadata of its own cluster as well as of their neighboring clusters. So this strategy will reduce efforts and time of proxy servers.

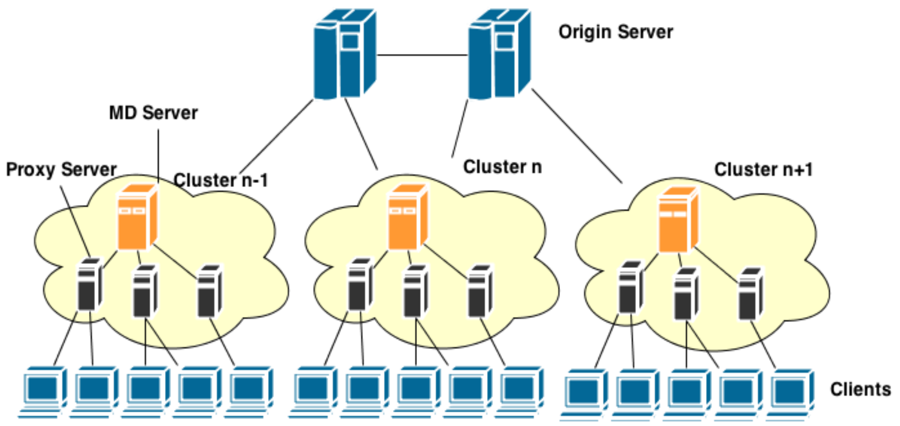

Figure 3: The Proposed Scheme's Architecture

How This Strategy Works:

Each proxy server's (PS's) metadata is denoted by md_{id} and each cluster's (CR's) metadata is denoted by MD_{id}. Each proxy server maintains its own metadata only. Metadata Server (MDS_n) in each $CLUSTER_n$ maintains 2 types of metadata, one is metadata of all proxy servers within that cluster (i.e. all md_{id} of cluster n) and secondly metadata of neighboring clusters (i.e. of MD_{n-1} and MD_{n+1}) as shown in Figure 4.

MD_{n-1}	MD_n	MD_{n+1}

Figure 4: Metadata maintained at MDS_n

Metadata Server at each cluster exchange their metadata with neighboring clusters periodically and all proxy servers within a cluster send their metadata (md_{id}) to MDS after every s seconds or whenever requested by the MDS. The time at which last metadata was sent is denoted by lmds. Also proxy servers keep sending their updated metadata (umd)

[21] to their MDS within s/m seconds where m-1 umds will be sent between lmds and lmds+s time. This will improve latency time as before serving a request, PS needs not to wait for the metadata exchange. Updated data will be available after every s/m seconds. All PS will send their md_{id} to MDS after every s seconds even if no umd is sent between lmds and lmds+s time to resolve the Staling issue as shown in Figure 5. Also MDSs at each cluster exchange their metadata with neighboring clusters periodically. In this scheme if any proxy server gets disconnected to the cluster there will be no issue of metadata management. After reconnection it already has its own metadata and metadata of other proxy servers of the same cluster and of neighboring clusters (i.e. MD_{n-1} and $MD_{n+1)}$ is maintained at the MDS. It will be immediately ready to listen to their clients request as before disconnection. This provides automatic and effective handling for frequent disconnections of proxy servers in the network. This scheme also benefits the PS by saving their time and efforts of metadata exchange with neighboring PSs. Also traffic within the cluster is reduced.

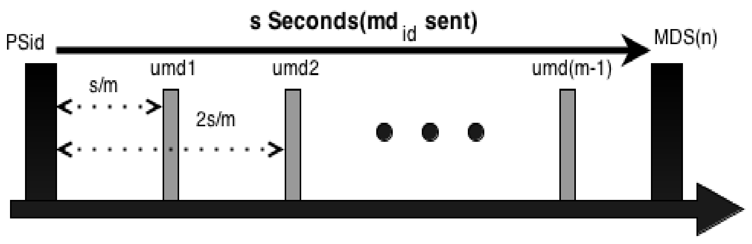

Figure 5: The md_{id} sent after every s seconds and umd after every s/m seconds to MDS

Phases for a Client's Request:

Every time a client made a request to proxy server the queue length of cluster is checked if it is under limit then queue length of proxy server is checked and if the client limit of proxy server has not exceeded, the client is served otherwise the request is forwarded to a less loaded proxy server. This makes efficient balancing of load in the network for proper handling of all client requests. This strategy will work in following possible phases whenever client requests a proxy server ($PS_{i)}$ of cluster n (CS_n) for some page:

1. After receiving request from the client the PS_i checks its own metadata md_i for the relevant page, if there is a Hit, the page is replied back to the client immediately.
2. In case of Miss in Step 1, the PS_i forwards request to the Metadata Server (MDS_n) of the same cluster. MDS_n will check metadata of the all other proxy servers fall into the same cluster (CS_n) for the requested page. If the page is found the request is forwarded to that proxy server and response is being transmitted back to the client.
3. If the page is not found in the other proxy servers of CS_n, the MDS_n will check its database for the metadata MD_{n-1} and MD_{n+1} of neighboring clusters CS_{n-1} and CS_{n+1} respectively for the requested page. If there is a Hit the request is forwarded to that PS of neighboring cluster and reply is being transmitted back to the client.
4. If the requested page is not found even in the neighboring clusters, the request is forwarded directly to the next neighboring clusters with a factor of 2 that are clusters CS_{n-2} and CS_{n+2}.
5. If the requested page is still not found, the request is forwarded directly to the origin server (OS). If there is a Hit at the origin server, the page is returned back to the client.
6. If the requested page is even not present at the origin server, a "Page Not Found" message is flashed back to the client.

These all possible phases for a client's request are also depicted in the sequence diagram shown in Figure 6 given below.

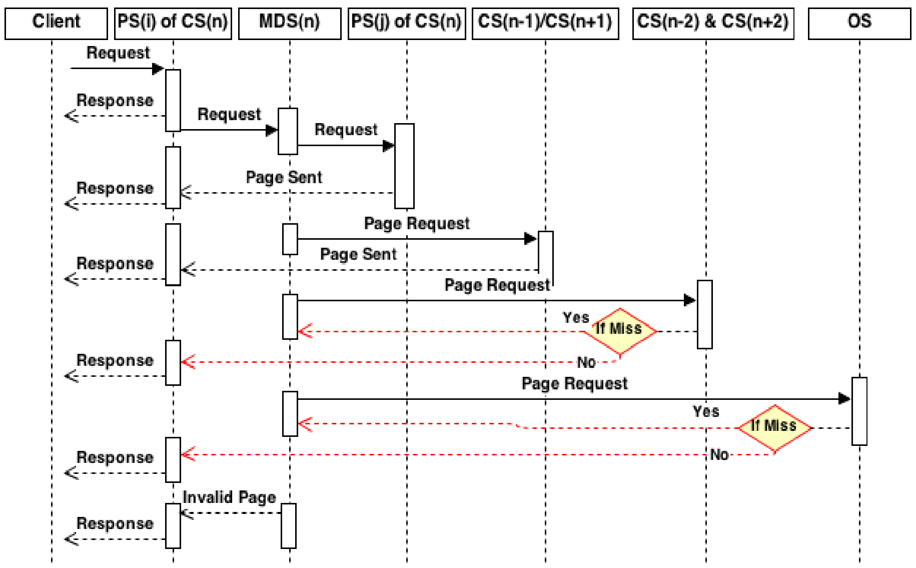

Figure 6: The sequence diagram showing all phases

This scheme can achieve low latency factor for a client's request as umds are sent to MDS after every s/m seconds. So in order to reply for a client's request proxy server never waits for any metadata exchange. Reply latency has reduced to s/m seconds. Also frequent disconnections of proxy servers are handled effectively. So now reconnected proxy servers can set back to their previous functionality easily. They start responding their clients and will send their updated list of registered pages i.e. umd at next s/m sec or at lmds+s time.

V. RESULTS AND DISCCUSSION

In this section we compare our strategy with earlier ones for evaluation parameters. We will compare our scheme with schemes proposed by Tiwari et al. (2010) DWC [19], Rajeev and Gulista et al. (2010) DWCC [20], Rajeev and Lalit et al. (2011) RDWC [9] and Rajeev and Neeraj et al. (2012) DWCRLD [3] on the basis of Scalability, metadata management, delays, latency etc. as given in table 1 below. As dynamic clustering scheme proposed by Rajeev Tiwari and Neeraj Kumar in [3] suffers from the limitations of scalability and management of metadata. Where every proxy server manages two types of metadata, one of all PS of same cluster and other is metadata of neighboring clusters. This scheme is not much scalable as if the number of proxy servers increased per cluster or if the size of the metadata grows per proxy server. This enhanced strategy can overcome these problems as MDS is managing all these issues. The more proxy servers can be added to the cluster without affecting the performance of the system. The metadata still remain manageable. The hit ratio is increased by a factor of 3. As MD of two neighboring clusters and of its own is maintained at MDS. Now PS needs not to bother about exchange of metadata with neighboring PSs and clusters as well. Along with this there is no exchange of metadata within a cluster. This also reduces the network traffic to some extent.

TABLE 1: COMPARISON OF PROPOSED SCHEME

Schemes	Hit Ratio	Delay	Scalability	Robustness	Meta Data Management	Latency	Disconnection Handling
DWC-2010 [19]	Average	Above Avg.	Low	Low	Low	High	No
DWCC-2010 [20]	Above Avg.	Low	Avg.	Avg.	High	High	No
RDWC-2011 [9]	Increase by factor k	Low	High	High	High	High	No
DWCRLD-2012 [3]	Better than RDWC-2011	Very Low	High	Better than RDWC	Same As RDWC	Very Low	Yes
Our Strategy-2013	Same as DWCRLD-2012	Very Low	Better than DWCRLD-2012	Same as DWCRLD-2012	Better than DWCRLD-2012	Same as DWCRLD-2012	Yes

VI. CONCLUSIONS

Web services have become very popular today, but server overloading, scalability, disconnections and network congestion etc. have become a tradeoff to the performance of web. Web caching has come up as a great solution for all these problems and issues. We have discussed some of the problems affecting the performance of web caching and major issues related with the distributed web caching. A number of caching schemes already exits. Some effective techniques such as DWC, DWCC, RDWC and DWCRLD schemes for distributed environment along with their limitations have been discussed in this paper. In this work, we have proposed a strategy called "*Enhanced Dynamic Web Caching: For Scalability & Metadata Management*" that can be easily deployed in the future. This is based on the DWCRLD to enhance the scalability and to alleviate extra overhead of metadata management of the proxy servers and also reduces the network traffic as well. This scheme also makes it easy to handle frequent disconnections in the network. By this even if the number of proxy servers grows in the network, metadata management will never be an issue. But this scheme is also having certain limitations such as hardware failure, cache routing, fault tolerance, proxy placement, security and dynamic data caching etc.

ACKNOWLEDGEMENT

We wish to express our deep sense of indebtedness and sincerest gratitude to all the experts and Mr. Rajeev Tiwari for his invaluable guidance and appreciation. We also wish to thank our Parents for their encouragement at every step of progress.

REFERENCES

[1] K. Claffy, H.W. Braun, *Web traffic characterization: An assessment of the impact of caching documents from NCSAs web server*, in Electronic Proc. 2nd World Wide Web Conf.94: Mosaic and the Web, 1994

[2] Pablo Rodriguez, Christian Spanner, and Ernst W. Biersack, *Analysis of Web Caching Architectures: Hierarchical and Distributed Caching*, IEEE/ACM Transactions On Networking, Vol. 9, NO. 4, AUG 2001

[3] Rajeev Tiwari, Neeraj Kumar, *Dynamic Web Caching: for Robustness, Low Latency & Disconnection Handling,* 2nd IEEE international conference on Parallel, Distributed and Grid Computing, 2012

[4] Chankhunthod et. al., *A hierarchical internet object cache*, in Proc. 1996 annual conference on USENIX Annual Technical Conference, San Diego, CA, Jan. 1996.

[5] Povey and J. Harrison, *A distributed Internet cache*, in Proc. 20th Australian Computer Science Conf., Sydney, Australia, Feb. 1997

[6] V. Cardellini, M. Colajanni, P.S. Yu, *Geographic Load balancing for scalable distributed Web systems*, Proc. of MASCOTS'2000, IEEE Computer Society, San Francisco, CA, pp 20-27 Aug. 2000.

[7] D.Wessels and K. Claffy, *Application of Internet cache protocol (ICP)*, version 2, Internet Engineering Task Force, Internet Draft: draft-wessels-icp-v2-appl-00. Work in Progress., May 1997.

[8] V. Valloppillil and K. W. Ross., *Cache Array Routing Protocol*, v1.1. Internet draft. [Online]. Available: http://ds1.internic.net/internetdrafts/draft-vinod-carp-v1-03.txt, 1998

[9] Rajeev Tiwari, Lalit Garg, *Robust Distributed Web Caching Scheme: A Dynamic Clustering Approach*, in International Journal of Engineering Science and Technology in ISSN : 0975-5462 Vol. 3 No. 2 Feb 2011,pp 1069-1076.

[10] Adam Dingle, Tomáš Pártl, *Web Cache Coherence, Journal reference: Computer Networks and ISDN Systems*, Volume 28, issues 7–11, p. 907.

[11] Problems with cache: *http://www.mnot.net/cache_docs/#META*

[12] Problems with cache: *http://www.webproworld.com/webmaster-forum/threads/71020-Website-Cache-Problem*

[13] S. Gadde, M. Rabinovich, and J. Chase, *Reduce, reuse, recycle: An approach to building large internet caches*, in *Proc. 6th Workshop on Hot Topics in Operating Systems (HotOS–VI)*, May 1997.

[14] P. Vixie and D. Wessels, *RFC 2756: Hyper text caching protocol*, (HTCP/0.0), Jan. 2000.

[15] R. Tewari, M. Dahlin, H. M. Vin, and J. S. Kay, *Beyond hierarchies: Design considerations for disturbed caching on the Internet*, in *Proc. ICDCS '99 Conf.*, Austin, TX, May 1999.

[16] A. Rousskov and D. Wessels, *Cache digest*, in *Proc. 3rd Int. WWW Caching Workshop*, June 1998, pp. 272–273.

[17] Makpangou, G. Pierre, C. Khoury, and N. Dorta, *Replicated directory service for weakly consistent replicated caches*, in *Proc. ICDCS'99 Conf.*, Austin, TX, May.

[18] Rabinovich, J. Chase, and S. Gadde, *Not all hits are created equal: Cooperative proxy caching over a wide-area network*, in *Proc. 3rd Int. WWW Caching Workshop*, Manchester, U.K., June 1998.

[19] Tiwari Rajeev and Khan Gulista, *Load Balancing in Distributed Web Caching: A Novel Clustering Approach*, Proc. of ICM2ST-1O,International Conference on Methods and models in science and technology pp. 341-345, November 6, 2010, vol.l324

[20] Rajeev Tiwari, Gulista khan, Load *Balancing through distributed Web Caching with clusters*, Proceeding of the CSNA 2010 Springer, pp 46-54, Chennai, India.

[21] G. Cao, *On Improving the Performance of Cache Invalidation in Mobile Environments*, Mobile Networks and Applications 7, 291–303, 2002.