DYNAMIC DISTRIBUTED WEB CACHING WITH KNOWLEDGE BASED CLUSTERING

Namit Gupta [1], Rajeev Kumar and Gulista Khan [2]

[1,2]Computer Sc. & Engg. Department, Teerthanker Mahaveer University,
Moradabad, Uttar Pradesh, India
namit.k.gupta.coe@tmu.ac.in, rajeev2009mca@gmail.com

Abstract: The Distributed Web Caching System suffers from scalability and less robustness problem due to overloaded and congested proxy servers. Load Balancing and Clustering of proxy servers helps in fast retrieval of pages, but cannot ensure robustness of system. In this paper we have given solution for scalability and robustness of Distributed web caching System and for load balancing Clustering and metadata manageability. We have also refined our technique using proxy server clusters with knowledge based Clustering and dynamic allocation of requests. We devised an algorithm for Distributed Web Cache concepts with knowledge based clusters of proxy server based on geographical regions. It increases the scalability by maintaining metadata of neighbors. We are making clusters based on knowledge proxy serves having similar data are collectively make a cluster. Based on which hit ration will be high. It increases the scalability by maintaining metadata of neighbors collectively and balances load of proxy servers dynamically to other less congested proxy servers, so system doesn't get down unless all proxy servers are fully loaded so higher robustness of system is achieved. This algorithm also guarantees data consistency between the original server object and the proxy cache objects using semaphore.
[Namit Gupta, Rajeev Kumar, Gulista Khan. **Dynamic Distributed Web Caching with Knowlwdge Based Clustering.** Researcher. 2012;4(1):30-36]. (ISSN: 1553-9865). http://www.sciencepub.net.

Keywords: Distributed Web Caching; Knowledge based Clustering; Proxy Server; Latency; Hit Ratio; Metadata; Robustness.

1. INTRODUCTION

As the World Wide Web (WWW) is gaining more and more popularity, servers have to handle more requests accordingly. The more people (or simply clients) request resources (in this case files) from web servers, the faster servers have to accept and process the requests. To cope with these requirements programmers as well as system administrators must take countermeasures. From the very beginning of the WWW the requirements for servers have not only changed from the view of traffic, but also from the type of content they deliver to the client. Initially static pages had to be served, today in 2005 content is usually taken from a database, and dynamically generated pages are to be transferred. This development takes the main source of load away from the operating system responsible for reading the files from the hard disk or another type of memory and shifts it to the program that dynamically generates the page. Also computer hardware has evolved. This makes it possible to have web pages generated the way they are today. Generally speaking, servers are capable of serving most pages in quite a reasonable amount of time. This is true as long as only a small number of visitors request pages to be generated. The larger the numbers of clients, the more pages have to be generated simultaneously. Multi-tasking enables servers to do so, but CPU capacity is limited. If it was only for system administrators, they would add more hardware power (for instance clustering servers, load balancing). Often this can be done only to a certain extent, mainly due to financial but also for logistical reasons. From a programmer's view, however, algorithms can be optimized (consider an algorithm in O(n2) on a fast computer which can easily be overtaken by a slower one running an O(n)) but also by caching techniques. The basis for this diploma thesis will be the analysis of caching strategies for this scenario. They will be used to speed up an existing application. The combination of various methods will be tested and benchmarked to reach a stage at which the application runs at reasonable speed even under high load.

Most popular web sites are suffering from server congestion, since they are getting thousands of requests every second in coincidence or not with special events. Moreover, the heterogeneity and complexity of services and applications provided by web server systems is continuously increasing. Traditional web publishing sites with most static contents have being integrated with recent web commerce and transactional sites combining dynamic and secure services. The most obvious way to cope with growing service demand and application complexity is adding hardware resources because replacing an existing machine with a faster model provides only temporary relief from server overload. Furthermore, the number of requests per second that a single server machine can handle is limited and cannot scale up with the demand. Two common approaches to implement a large scale cache cooperation scheme are hierarchical and distributed [1],

[2] caching. The need to optimize the performance of Web services is producing a variety of novel architectures. Geographically distributed web servers [3] and proxy server systems aim to decrease user latency time through network access redistribution and reduction of amount of data transferred, respectively. In this it consider different web systems, namely web clusters that use a tightly coupled distributed architecture. Cluster technology is a cost effective solution because it is cheaper than a single faster machine. From the user's point of view, any request to a Web cluster is presented to a logical server that acts as a representative for the Web site. This component called Web switch retains transparency of the parallel architecture for the user, guarantees backward compatibility with Internet protocols and standards, and distributes all client requests to the Web and back-end servers. Cluster architectures with Web switch dispatcher have been adopted with different solutions in various academic and commercial Web clusters. A valuable recent survey is in [4].One of main operational aspects of any distributed system is the availability of a mechanism that shares the load over the server nodes. Numerous global scheduling algorithms were proposed for multi-node architectures executing parallel or distributed applications. Unlike traditional distributed systems, a Web cluster is subject to quite different workload. The hit rate of a Web cache can be increased significantly by sharing the interests of a larger community [5]; the more people are accessing the same cache, the higher the probability that a given document is present in the cache. To increase the effective client population using a cache, several caches can cooperate.

2. Problems in Distributed web caching [6]

2.1 Extra Overhead

Extra Overload increases when all the proxy servers keep the records of all the other proxy servers which results congestion on all proxy servers. They all have to keep check on the validity of their data which results in extra overhead on proxy servers.

2.2 Size of Cache

If Cache Size is large then Meta data become unmanageable because in traditional architectures each proxy server keeps records for data of all other proxy servers. In this way if Cache size becomes large then maintenance of Meta data is a problem.

2.3 Cache Coherence Problem

When client send requests for data to proxy server that data should be up-to-date. This results into Cache Coherence problem.

2.4 Scalability

Finally, By Clustering we can also solve the problem of scalability, add more number of clients, and data of these clients will be managed on the basis of geographical region based Cluster. A particular cluster will only have to manage the id's and update the Meta data of all the proxy servers or of the neighbour clusters.

2.5 Robustness

In distributed web caching there is no any concept of clustering so the mostly user is not satisfied because they request to same proxy server so the system is no more robust.

2.6 Hit Ratio

When the congestion will occur in the network then the hit ratio will be decreased because all the clients will wait for requested pages.

2.7 Load balancing

When there is no any limit to connect the client to proxy server then most of the clients will connect to same proxy server. Means one proxy server will busy and other proxy server will free.

3. Proposed Solution for Distributed Web Caching

Extra overhead on proxy server, problem of unmanageable data, Cache coherence, less robustness and Scalability. Extra Overhead increases when all the proxy servers keep the records of all the other proxy servers which results congestion on all proxy servers. They all have to keep check on the validity of their data which results in extra overhead on proxy servers. Clustering reduces this extra overhead. By making clusters on the basis of Geographical region we can solve this problem. From now one proxy will manage the Meta data regarding the proxy servers which fall under the same cluster region and its neighbour clusters. By using the concept of Knowledge based clustering more clients request will be satisfied then the system will be more robust. If Cache Size is large then Meta data become unmanageable because in traditional architectures each proxy server keeps records for data of all other proxy servers. In this way if Cache size becomes large then maintenance of Meta data is a problem. When client send requests for data to proxy server that data should be up-to-date. This results into Cache Coherence problem. This can be solved by having a timer with origin server, after a particular time period if there is any fresh page then origin server will check for it and send the fresh pages to any of proxy server which in turn forward this information to all other proxies and one cluster region have to maintain the record for cache updation for that cluster only. This gives proper updation of data and also less workload. With all the data pages there will be a time stamp field,

now if there is any fresh page then origin server will check for it and send the fresh pages to any of proxy server which in turn forward this information to all other proxies and one cluster region have to maintain the record for cache updation for that cluster only. This gives proper updation of data and also less workload. Finally, By Clustering we can also solve the problem of scalability by this, add more number of clients, and data of these clients will be managed on the basis of geographical region Cluster. A particular cluster will only have to manage the id¡¦s and update the metadata of all the proxy servers fall under the, particular geographical region cluster and its neighbour clusters.

4. The Proposed Caching System

In this section, we first define the structure for which our proposed caching model is intended. This is followed by a detailed description of the caching model. We then discuss some interesting properties of the proposed system.

4.1. Structure of Network

At the highest level origin server are scattered around the world. These origin servers are connected to proxy server via various form of communication medium. These proxy servers are in turn connected to client computer whose purpose are just to send the request and get response. Proxy servers are arranged at middle level which acts as both client and servers. For origin server they act as client and for client computers they act as servers. All the proxy servers are arranged in the geographical region based. After adding the concept of clusters now it is easy to maintain the data in the cache of proxy servers. To maintain the data consistency we associate a timer with the origin server so that if there is any fresh page then origin server keep check for that and forward fresh pages to the proxy servers on designated port which in turn forward data to other proxy servers to maintain metadata. Each Proxy server is having metadata in its cache which keeps the record about data in other proxy servers. Each proxy server in one cluster is having metadata about proxy servers which fall in corresponding cluster and proxy servers of neighbor clusters.

4.2. Proposed Architecture

According to the architecture described in [7] browsers are at the lower level, proxy servers at the middle level and origin servers at the highest level.

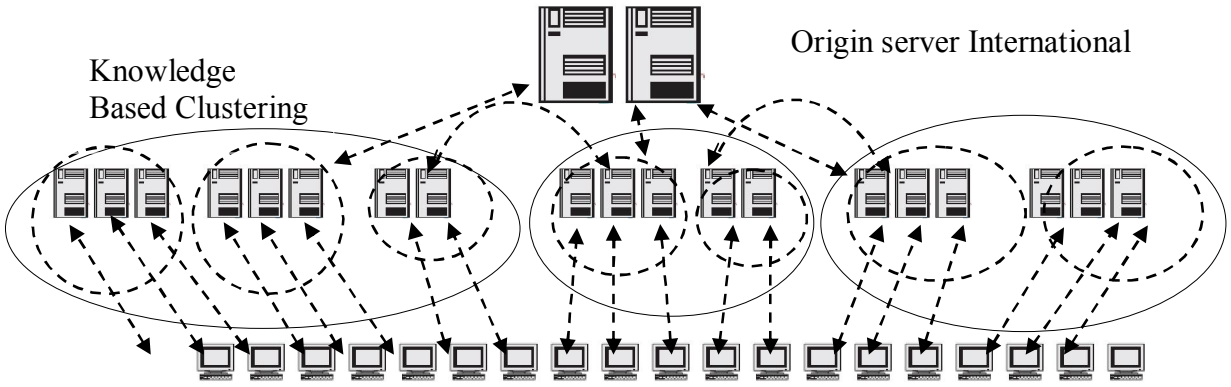

Fig.1 Proposed Architecture of Knowledge based clustering

4.2.1. Working

These problems like extra overhead problem of unmanageable data, cache coherence, less robustness and scalability are major problem for data retrieval from proxy servers. In some earlier paper solutions for these problems are given by making clusters on geographical region based. We get a high benefit by clustering but in the term of hit ration, if we want to increase hit ration we can add the concept of knowledge based clustering. In this technique, knowledge based clusters are formed based on similar knowledge for example, if cluster become on similar knowledge in the same region, like is in same city requirement is for computer defined data then all the cluster fall in that region will be rich in data regarding computer.

Whenever a query found from a client to proxy server for a particular data, firstly queuelength check as earlier defined in [3] after that pattern match in this proxy server if pattern match in then same knowledge based pattern cluster are picked and page is reached in that clusters metadata, if data found then ok otherwise data is requested to origin server.

4.3. Proposed Algorithms

We have given the Algorithm for Proxy servers.

4.3.1. ProxyServers:

/* queuelength: It is associated with every proxy server, which tells how many client's requests can be made to a proxy server.*/

- PS: Proxy Servers.
- Noofservices: tells how many connections are active with proxy server.
- CIP: Client's Internet Address.
- Reply: has either requested page or message "NO".
- Rpage: Requested page or file.
- Ps_ip []: is a stack of Internet Addresses of all the proxy servers in same Cluster.
- OS []: is a stack of Internet Addresses of all the Origin Servers.
- Cluster []: is a stack of Internet Addresses of all Clusters.
- MB: Match Bit
- KBC: Match Bit for Knowledge based clustering
- KBC[]: is a stack of Internet Addresses of all the proxy servers in same Knowledge based clustering
- KBC_CSE[]: is a array of computer science Engineering
- KBC_ECE[]: is a array of Electronic & communication Engineering
- KBC_ME[]: is a array of Mechanical Engineering
- KBC_CVE[]: is a array of Civil Engineering

Step 1: queuelegnth(ql)=0;
Step 2: If Request from OS [] for connection then
2.1 Establish connection with the origin server.
2.2 Connection Established.
(1) A new thread is established.
(2) Receive data from origin server.
(3) Update its metadata and broadcast information to all clusters.
Step 3: Proxy Server will wait for connection with clients or from other proxy servers.
Step 4: If request is to update data by any other proxy server then update metadata.
Step 5: if (q11+q12+q13) <=240
Step6: if (q1<80)
Step7: A Connection is established by creating a new thread to deal with it & client's locative address in CIP.
Step8: Get client's locative on in IP address in CIP.
Step9: if Incoming request from Client CIP /*request is from client*/

(a) If Pattern match is on same KBC [] /*KBC_CSE[], KBC_ECE[], KBC_ME[], KBC_CVE[] */
9.1 wait();
9.2 ql=ql+1;
9.3 Signal();
9.4 Search metadata();
9.5 If matchfound then
(1) If matchfound is on current cluster
1. If (same proxy server)
i. wait();
ii. ql=ql+1;
iii. signal();
iv. Search for Rpage in cache.
v. Return Rpage to CIP.
vi. ql=ql-1;
(2) else/*else of 1.*/
1. Send request to another proxy server.
2. looking for reply.
3. Return Rpage to CIP.
9.6 else /*else of (1) */
(1) if(MB==1)/* Data on Neighbour cluster*/
1. wait();

2. ql=ql+1;
3. signal();
4. Return Rpage to CIP.
5. ql=ql-1;

(2) else If (MB==0)/*else of (1) */ /*Increment cluster to search*/

1. (Owncluster+2)% n
2. If (r page=NO)
   - No Such page Exist in This Cluster & Goto step A
3. else /* Else of (2).2*/
   i. Wait();
   ii. Ql=ql+1;
   iii. Signal();
   iv. Return Rpage to CIP.
   v. Ql=ql-1;

else /*else of (a) */

(1) if(KBC==1)/* Data on Neighbor knowledge based cluster*/

1. wait();
2. ql=ql+1;
3. signal();
4. Repeat step 9
5. Return Rpage to CIP.
6. ql=ql-1;

(2) else If (KBC==0)/*else of (1) */ /*Increment knowledge based cluster to search*/

1. (Ownknowledgecluster+2)% n
2. If (r page=NO)
   - No Such page Exist in This Cluster & Goto step A
3. else /* Else of (2).2*/
   i. Wait();
   ii. Ql=ql+1;
   iii. Signal();
   iv. Repeat step 9
   v. Return Rpage to CIP.
   vi. Ql=ql-1;

Step 10: Else/*else of step 9*/

10.1. If(request is from proxyserver)&&(MB==1)&&(Same Cluster)
(1) Accept Connection();
(2) wait();
(3) ql=ql+1;
(4) signal();
(5) Return Rpage to Proxy Server.
(6) ql=ql-1;

10.2 .Else If(request is from proxyserver)&&(MB==1)&&(Different Cluster)/*else of 10.1*/
(1) Accept Connection();
(2) wait();
(3) ql=ql+1;
(4) signal ();
(5) Return rpage to proxy server.
(6) ql=ql-1;

10.3 Else if (request from proxy server) && (MB==0)/* Else of 10.2*/
(1) Accept Connection ();
(2) wait ();
(3) ql=ql+1;
(4) signal();
(5) Search Metadata();
(6) if(DataFound)

(7) GoTo Step 9.5
10.4 Else /*else of 10.3.(6)*/
(1) Send Request to Origin server.
(2) Accept Connection from origin server.
(3) If datafound
- Send Rpage to proxy server.

(4) Else /* Else of 10.4.(3)*/
- B. No page exist.

Step 11. If (Request is from origin server)
11.1. Accept Connection();
1.2. Update metadata & cache;
Step 12 Else If(ps2.ql<80)/*Else of step 6*/
12.1. Forward request of client to ps2.
Step 13. Else Forward request of client to ps3. /*Else of step 12*/
Step 14. Else Send Request of client to any other cluster. /*Else of step 5*/

4.3.2 Explanation

In this Algorithm , Firstly request goes to Proxy server from Client then proxy server will check queue length of cluster in which that proxy server lies , if it is less than 240(we have fixed the queuelength of proxy server say 80 ,and consider three proxy server in a cluster) then check the queue length of proxy server if it is less than 80 then check that the request is from client or from other proxy server if request is from client then hold the wait signal for lock and increase its queue length by 1 if requested page is found in its current knowledge based cluster and on current cluster on same proxy server then return Rpage to client otherwise check its metadata it has all information of its own proxy servers and all information of its neighbor clusters. If it is found in neighbor cluster then there is an match bit (MB) it is equal to one it means that proxy server know the requested page in that proxy server because it has all information of its neighbor proxy server send request to that match proxy server otherwise send request to its neighbor cluster and match bit is equal to zero if it has requested page then send reply back to that proxy server otherwise it can be on different cluster. If it is found in neighbor knowledge based cluster then there is an Match Bit for Knowledge based clustering (KBC) it is equal to one it means that proxy server know the requested page in that proxy server because it has all information of its neighbor proxy server send request to that match proxy server otherwise Repeat the previous processing otherwise send request to its neighbor cluster and Match Bit for Knowledge based clustering is equal to zero if it has requested page then send reply back to that proxy server otherwise it can be on different knowledge based cluster. If request is from proxy server of its same cluster and MB is equal to one then return Rpage to that proxy server if request is from different cluster and MB equal to one then check in its metadata and return page to the proxy server else there is an third case request is from proxy server and MB is equal to zero it means that requesting proxy server don't have the information of that proxy server. If the requested page is not found in all the proxy servers then send the request to the origin server and make connection with origin server and receive the requested page. There is another case the request can from origin server to update the information in metadata than proxy server will receive the updated copy from origin server and will update it.When the client will request to the proxy server there may be possibility that the proxy server have no more space means it is equal to its defined queue length then send request to other proxy server in same cluster if no space then send other proxy server otherwise send the request to the other cluster.

Working of Client is just to send request for Rpage to the Proxy server and wait for result if result comes under the time limit then process the response otherwise send request again.Working of Origin Server is to send the updated data after a particular time interval say after 3sec., and send the requested pages to proxy servers whenever request comes.

5. Results and Discussion

Based on this algorithm we got the better results than previous algorithms in terms of hit ratio, delay, scalability and robustness.

In this table there is comparison between three architectures. In distributed web caching the hit ratio was Average and delay was Above Average and scalability, robustness was low, unmanageable data was high. In second Architecture the result was better than previous in case of hit ratio, Delay, scalability and robustness. And in third Architecture the results are better than previous architectures in case of hit ratio, scalability and robustness.

Table-1 Comparison between Different Architectures

Network	Hit Ratio	Delay	Scalability	Robustness	Unmanageable MetaData
Distributed web Caching	**Average**	**Above Average**	**Low**	**Low**	**High**
Distributed web caching with clustering	**Above Average**	**Low**	**Average**	**Average**	**Low**
Knowledge based clustering distributed web caching	**High**	**Low**	**High**	**High**	**Low**

6. Conclusions

Web service becomes more and more popular, users are suffering network congestion and server overloading. Great efforts have been made to improve Web performance. Web caching is recognized to be one of the effective techniques to alleviate server bottleneck and reduce network traffic, thereby minimize the user access latency. In this work, I give an algorithm to reduce the Extra Overhead, solves the problem of Cache Coherence (Get if Modified), problem of Scalability along with solving all these problems it also improves the Hit Ratio and the Latency Time .By surveying previous works on Web caching, we notice that there are still some open problems in Web caching such as proxy placement, cache routing, dynamic data caching, fault tolerance, security, etc. The research frontier in Web performance improvement lies in developing efficient, scalable, robust, adaptive, and stable Web caching scheme that can be easily deployed in current and future network. Further we can improve performance by adding page replacement algorithm and by making it pure Dynamic.

REFERENCES

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

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

[3] 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.

[4] T. Schroeder, S. Goddard, B. Ramamurthy, "Scalable Web server clustering technologies", IEEE Network, May-June 2000, pp. 38-45.

[5] K. Claffy and H.W. Braun, "Web traffic characterization: An assessment of the impact of caching documents from NCSA"s web server" in Electronic Proc. 2nd World Wide Web Conf.'94:pp- 37-51 Mosaic and the Web, 17-10-1994 vol.28.

[6] Tiwari Rajeev and Khan Gulista, "Load Balancing in Distributed Web Caching: A Novel Clustering Approach", Proc. of ICM2ST-10, International Conference on Methods and models in science and technology pp. 341-345, November 6, 2010, vol.1324.

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

12/12/2011