ISBN 978-93-5156-328-0
International Conference of Advance Research and Innovation (ICARI-2014)

# Semantic Web System using Web Caching Algorithm at Origin Server for different Web Services

Rajeev Kumar *, Hina Hashmi
Department of Computer Application, TMU, Moradabad, Uttar Pradesh, India

**Article Info**

Article history:
Received 2 January 2014
Received in revised form
10 January 2014
Accepted 20 January 2014
Available online 1 February 2014

**Keywords**

Distributed web caching,
Origin server algorithm,
Web services,
ICT,
Java Application

**Abstract**

In this paper we will discuss that today's the most popular web sites are suffering from the server congestion, and they are getting thousands of requests every second from the client. And in this regarding design a origin server algorithm whose manage the services. 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 as dynamic and secure by services. The most understandable 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. The need to optimize the performance of Web services is producing a variety of novel architectures.

## 1. Introduction

The quality of service and the response times can be improved by decreasing the network load. One way to achieve this is to install a Web caching service. Caching effectively migrates copies of popular documents from Web servers closer to the Web clients. In general, Web client users see shorter delays when requesting a URL, network managers see less traffic and Web servers see lower request rates. An origin Web server might not only see lower requests rates but primarily will experience a lower server load because files will be fetched with an If-Modified-Since GET HTTP request. Web clients request documents from distributed Web servers, either directly or through a distributed Web cache server system or proxy. A Web cache server has the same functionality as a Web server, when seen from the client and the same functionality as a client when seen from a Web server. The primary function of a Web cache server is to store Web documents close to the user, to avoid pulling the same document several times over the same connection, reduce download time and create fewer loads on remote servers.

However, a hierarchical caching architecture needs powerful intermediate caches or intelligent load-balancing algorithms to avoid high peaks of load in the caches that will result in high client latency. In other hand, the distributed web caching has very good performance for all services in well-interconnected areas without requiring any intermediate cache levels. Nevertheless, the deployment of distributed caching on a large scale encounters several problems, such as large network distances, high bandwidth usage, and administrative issues.

**Corresponding Author,**
**E-mail address:** rajeev2009mca@gmail.com


### 1.2 Web Tier

The Web tier of Java EE application server is responsible for interacting with the end user such as web browsers primarily in the forms of HTTP requests and responses. And at the highest level cache, the Web tier does four basic tasks:

- Interprets client requests
- Dispatches those requests to an object that encapsulates business logic
- Selects the next view for display
- Generates and delivers the next view

The Web tier receives each incoming HTTP request and invokes the requested business logic operation in the application. In distributed Web Cache is a content-aware server accelerator, or reverse proxy server, for the Web tier that improves the performance

of the web services, scalability, and availability of Web sites on the server that run on HTTP Server. [1]

## 2. Web Caching Architectures

The performance of a Web cache system depends on the size of its client community; the bigger is the user community, the higher is the probability that a cached document will soon be requested again. Caches sharing mutual trust may assist each other to increase the hit rate. A distributed web caching architecture should provide the paradigm for proxies to cooperate efficiently with each other. [2]

### 2.1 Distributed dynamic Web Services using Caching and Clustering Support

Caching dynamic pages at a server site is beneficial in reducing server resource demands and it also helps dynamic page caching to the proxy sites. And such an approach can be cumbersome or inefficient in dealing with an arbitrarily large number of distributed pages. [4] We study techniques for partitioning dynamic pages into classes based on URL patterns. Our scheme allows an application to specify page identification and data dependence for a class of dynamic pages and invalidate them collectively. A data structure is developed for efficient URL class searching during lazy or eager invalidation. [9] We also implemented caching software called Cachuma which integrates the techniques, and runs in tandem with standard Web servers to the clients, and allows to Web services to add distributed dynamic web pages caching capability with minimal changes. [3]

### 2.2 Peer to Peer Dynamic Clusters

N-Cache provides a dynamic clustering capability with 100% uptime for the cluster. In this due to a peer to peer architecture of the particular cluster where there is no single point of failure.

A cache cluster is a collection of one or more cache servers with every server connected to every other server in the cluster. When a cache cluster is in formed, it contains a particular cluster coordinator that manages all membership to the cluster. To the coordinator is the oldest server in the cluster. If in the coordinator ever goes down, then this role passes on to the next senior-most server in the cluster. Here this removes any single point of failure in cluster membership management system. [6]

### 2.3 Hybrid Caching Architecture

In a hybrid scheme, caches may cooperate with other caches at the same level or at a higher level using distributed caching. ICP [7] is a typical

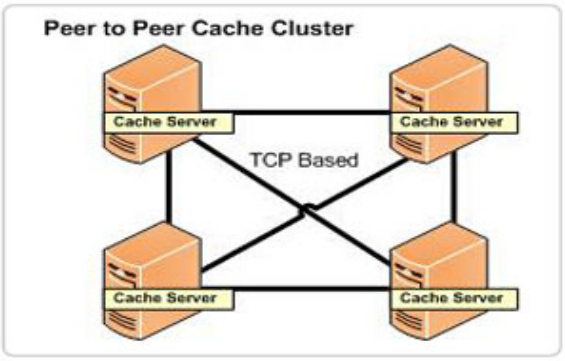


**Fig:** 1. Peer to Peer cache Cluster [6]

example. The document is fetched from a parent/neighbor cache that has the lowest RTT. Rabinovich et al. [8] proposed to limit the cooperation between neighbor caches to avoid obtaining documents from distant or slower caches, which could have been retrieved directly from the origin server at a lower cost.

## 3. Distributed Dynamic Clusters For Web Caching Services

Working of this architecture is like whenever any client request for any Page that request goes to the proxy server if the maximum queue length is not achieved yet then proxy server process that request search for this page in the cache [11] if page found their than response will generated from that place otherwise the page is searched in the metadata of that proxy server, if page found their request is forwarded to that proxy server in the same cluster.

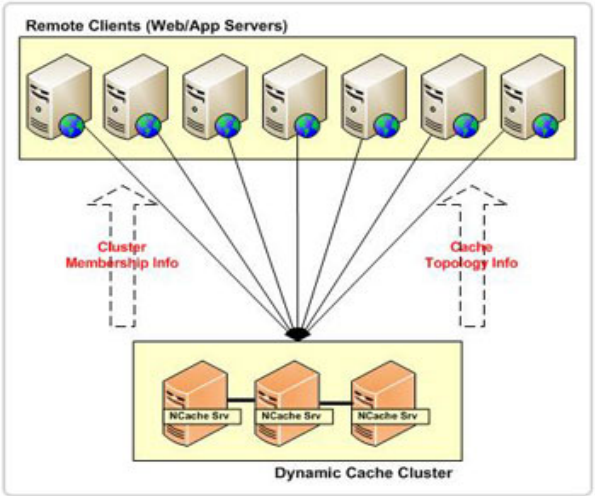


**Fig: 2.** Dynamic Cache Clusters [6]

## 4. Distributed Web Caching Algorithm For Origin Servers

/* Whole data files resides at origin server's secondary memory at a fixed directory path in directory */

**Step 1**: Start timer / time_counter

**Step 2**: If(time_counter>=3min) then

**Step2.a**: Create metadata of updated pages.

**Step2.b:** Check and list the updated pages.

**Step3:** Try to establish connection with a proxy server.

**Step 3.a:** Connection established.

**Step 3.b:** Send metadata to proxy server.

**Step 4:** Wait for the connection with proxy servers at a specified port.

**Step 5:** Connection is established.

**Step 5.a:** A new thread is created to deal with proxy server's connection with Origin server.

**Step 5.b:** As proxy server makes a request for a page then:

i. Fixed directory path is searched for requested page or file.
ii. If page exists then
iii. Desired file/page is returned to proxy server
iv. Else "No" is returned.

**Step 6:** Close (Exit).

## 5. Java Application for Distributed Web Caching Algorithm

For this algorithm is implementation using the concept of Socket Programming in JAVA. In it Client algorithm is simply make a connection with this a particular proxy server and request for a page and starts a timer. If response comes within time then ok otherwise it will try again and again. In Proxy server implementation there is a class[10] which will invoke every time when a new thread is generate or a new request comes .Depending on the request for Client , neighboring Proxy servers or Origin sever different functions executes. Proxy cache is implemented using Vector Arrays.

## 6. Challenges Of Distributed Web Caching System

In distributed web caching system, documents can be cached at the clients, the proxies, and the servers. A client always requests page from its local proxy if it doesn't have a valid copy of such page in its own browser's cache. Upon receiving a request from the client side, then the proxy first checks to see if it has the requested pages. So, it revert the page to the client. [12] If it doesn't have the requested page in its cache, it sends a request to its cooperative proxies or the server.

## 7. Result Discussion

In Our performance of evaluation methodology is that the firstly explained performance metrics and then based on this algorithm at origin server for different Cache hierarchies. in distributed Web cache system performance includes the throughput, mean response time, miss response time, hit response time, hit ratio, error rate, queuing of requests, connection length etc.

## 8. Conclusion

Today as Web service becomes more and more popular, and users are suffering network congestion problems and server overloading problems. Great efforts have been made to improve the web services performance. In this distributed web caching is recognized to be one of the effective techniques to alleviate server bottleneck and reduce network traffic problem, thereby minimize the user access latency. Here we design a distributed web caching system algorithm at origin server to make a more robust system and to reduce the Extra Overhead of the system solves the problem of Cache Coherence, problem of Scalability along with solving all these problems it also improves the Hit Ratio and the Latency Time using the java application.

## References

[1] http://docs.oracle.com/cd/E27559_01/doc.1112/e28391/ha_webtier.htm.
[2] Rousskov, "On performance of caching proxies," in Proc. ACM SIGMETRICS, Madison, WI, Sept. 1998.
[3] http://www.cs.ucsb.edu/projects/swala/
[4] Huican Zhu, Hong Tang and Tao Yang, Demand driven Service Differentiation for Cluster-based Network Servers**.** Technical
[5] Report #TRCS00-19, Dept. of Computer Science, UCSB, July 2000.Postscript version. To appear in IEEE INFOCOM'2001
[6] Huican Zhu and Tao Yang, Cachuma: Class-based Cache Management for Dynamic Web Content. Technical Report #TRCS00-13, Dept. of Computer Science, UCSB, June 2000. Postscript version, to appear in IEEE INFOCOM'2001

[7] http://www.alachisoft.com/ncache/dynamic-clustering.html

[8] M. Baentsch, L. Baum, G. Molter, S. Rothkugel, and P. Sturm, "World Wide Web caching: The application-level view of the internet," IEEE Commun. Mag., pp. 170–178, June 1997.

[9] National Lab of Applied Network Research (NLANR). [Online], http://ircache.nlanr.net/

[10] J. Yin, L. Alvisi, M. Dahlin, and C. Lin. Using leases to support server-driven consistency in large-scale systems, In Proceedings of the 18th International Conference on Distributed Systems, May 1998

[11] P. Cao, J. Zhang, and K. Beach, Active cache: Caching dynamic contents on the web. In Proc. of the IFIP Intl. Conference on Distributed Systems Platforms and Open Distributed Processing, Sep 1998.

[12] A. Myers, J. Chuang, U. Hengartner, Y. Xie, W. Zhuang, and H. Zhang, A Secure, Publisher-Centric Web Caching Infrastructure, in IEEE INFOCOM 2001.

[13] J. Mogul and P. Leach, 1997, Simple Hit-Metering and UsageLimiting for HTTP, RFC 2227