Dynamic Web Caching Services of Different Java Application in e-Governance: Case Study

* Rajeev Kumar, ** Dr. M.K. Sharma

* Research Scholar, Department of Computer Science, Bhagwant University, Ajmer (Rajasthan)
Email ID: rajeev2009mca@gmail.com

** Department of Computer Science, Amrapali Institute, Haldwani. (Uttarakhand) – India
Email ID: sharmamkhld@gmail.com

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. 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.
[Rajeev Kumar, M.K. Sharma. **Dynamic Web Caching Services of Different Java Application in e-Governance: Case Study.** *Academia Arena* 2013;5(4):28-29] (ISSN 1553-992X). http://www.sciencepub.net/academia. 4

Keywords: web services, cloud communication, ICT, Java, Oracle, web caching, Dynamic Clusters.

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 Web servers, either directly or through a Web cache server 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. On the other hand, distributed caching has very good performance 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.

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. It is the outermost tier in the application server, closest to the end user. At the highest level, the Web tier does four basic tasks:

- Interprets client requests
- Dispatches those requests to an object (for example, an enterprise Java bean) 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. Oracle Web Cache is a content-aware server accelerator, or reverse proxy, for the Web tier that improves the performance, scalability, and availability of Web sites that run on Oracle HTTP Server. [1]

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 (previously requested) will soon be requested again. Caches sharing mutual trust may assist each other to increase the hit rate. A caching architecture should provide the paradigm for proxies to cooperate efficiently with each other. [2]

Scalable & 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 at proxy sites. Previous work has used fine-grain dependence graphs among individual dynamic pages and underlying data sets to enforce result consistency. Such an approach can be cumbersome or inefficient in dealing with an arbitrarily large number of dynamic 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. To make this scheme time-efficient with small space requirement, lazy invalidation is used to minimize slow disk accesses when identifications of dynamic pages are stored in memory with a digest format. Selective precomputing is further proposed to regenerate stale pages and smoothen load peaks. A data structure is developed for efficient URL class searching during lazy or eager invalidation. We also implemented caching software called Cachuma which integrates the above techniques, runs in tandem with standard Web servers, and allows Web sites to add dynamic page caching capability with minimal changes. Our experimental results show that the proposed techniques can efficiently handle class-based page invalidation and are effective in reducing server response times for tested applications. [3, 5]

Peer to Peer Dynamic Clusters

N-Cache provides a dynamic clustering capability with 100% uptime for the cluster. This is due to a peer to peer architecture of the 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 formed, it contains a cluster coordinator that manages all membership to the cluster. The coordinator is the oldest server in the cluster (meaning the first server that started). If the coordinator ever goes down, this role passes on to the next senior-most server in the cluster. This removes any single point of failure in cluster membership management. [6]

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 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.

Challenges of Dynamic clusters in 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 client, the proxy first checks to see if it has the requested page. If so, it returns the page to the client. If it doesn't have the requested page in its cache, it sends a request to its cooperative proxies or the server.

Conclusion

In this paper by using the concept of Clustering static or dynamic and Time Stamping we can solve the problems of extra overhead, Unmanageable data, Cache coherence Problem and the problem of scalability. We will implement the architecture in java using the concept of Servlets or JSP.

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 Report #TRCS00-19, Dept. of Computer Science, UCSB, July 2000.Postscript version. To appear in IEEE INFOCOM'2001.
5. 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.
6. http://www.alachisoft.com/ncache/dynamic-clustering.html.
7. www.google.co.in more helpful for searching the contents.