Volume 2, Issue 12, December 2012 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

# Performance Analysis of Web Caching Through Cache Replacement Based on User Behavior

**Anita Madaan**
CSE, Punjabi University
India

**Niraj Madaan**
CSE, Kurukshetra University
India

**Poonam Jain**
CMJ University
India

***Abstract -- The Increasing growth of user over world wide web degrades the performance of system by network traffic and server heavy work load. To address this problem web caching is more efficient and scalable scheme to distribute web documents over the web. It is used in large distribution system and sits between web server and many clients. By introducing the web caching concept we maintain the caching system on our machine. Client sends the Http request by Specifying the URL. Redirector directs the client request to appropriate cache on cache cluster. It fetches the static contents from the original web page. The response with static content gets stored in user made my caching system. Next time, when user sends the same request it does not go to original server first, It checks in caching. LRU cache replacement use to update caching system when the original server updates the documents that resided in our cache. In order to maximize the hit rate we have to maximize the number of requests that have been satisfied in the cache. It depends on cache size as the cache size increases more hit rates get. When the hit rates maximize users get the results with minimum delay. The caching system finally upgrades the performance. It reduces the traffic and improves the access latency.***

***Keywords – World Wide Web, Cache, Sharing.***

## I. INTRODUCTION

As the number of World Wide Web users grows this increases both network load and server load. These scalability problems can be faced on three fronts. The first is to scale web servers to handle the increasing demands. The second is to ensure that the internet itself will scale by continuing to increase its capacity and by deploying new network technologies. The third approach is a more efficient and scalable schemes to distribute web documents in the internet.

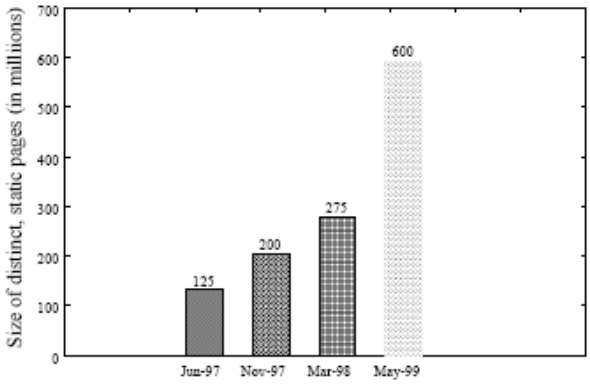



*Fig 1. Size of distinct static web pages*

World wide web also has a lot of documents that appeal to a wide range of interests e.g education, news, sports, scientific research, entertainment, stock market growth, travel , shopping, weather maps etc. Although the internet backbone capacity as increases as 60 % per year , the demand for bandwidth is likely to outstrip supply for the future as more and more information services are moved on to the web. As traffic on the web increases, users are faced with increasing delays which results of increase in the user perceived latency. Potential resources of the latency are the web servers heavy load, network congestion, low bandwidth, bandwidth under utilization and propagation delay. Nevertheless the user expects a high quality of service with modest response time. if some kind of solution is not undertaken for the problems caused by web rapid increasing growth, then www would become too congested and its entire appeal would eventually be lost. This problem can

be solved by using the technique of web caching in our system. Web caching has introduced as an effective solution to the problems of traffic congestion and effective bandwidth utilization over the web.

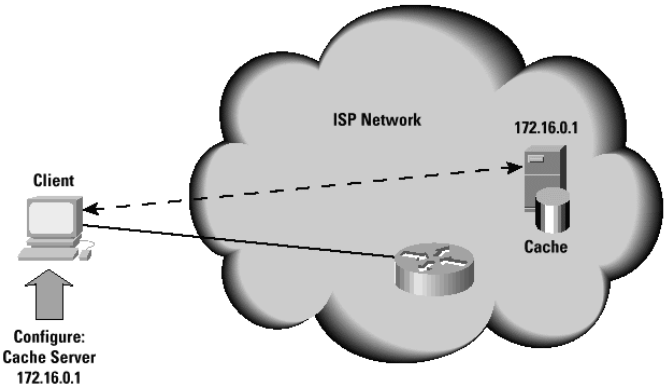



Fig 1.2 A Generic model of Web Cache

the requested page . If so, it returns the page to the client. If it does not have the requested page in its cache then it sends a request to its cooperative proxies or the server. Upon receiving a request from another proxy, a proxy checks if it has the requested page .If so, It returns the page to the requesting proxy. If not, the proxy may further forward the request to other proxies or the server. If none of the cooperative proxies has such page the requested page fetch from the server.

*A. Cache Replacement Policy*

A cache server has a fixed amount of storage for storing objects. When this storage space is full, the cache must remove some objects in order to make room for newly requested objects. The cache replacement policy determines which objects should be removed from the cache. The goal of the replacement policy is to make the best use of available resources, such as disk, memory space and network bandwidth. Since web use is the dominant cause of network backbone traffic today, the choice of cache replacement policies can have a significant impact on global network traffic. Many different factors to be considered while making cache replacement policy.

- **Live Documents**: Live documents are a small fraction of all documents. The cache only needs to retain live documents to achieve the maximum hit rate.
- **Inter access time:** Inter access time is the time between successive document requests. Documents having lower inter access times are the documents that are more likely to be requested in the future. Due to always selecting the document with the largest inter access time to be evicted.
- **Number of Previous Accesses:** Using the number of previous accesses made to a document is a good indication. We can use it to evaluate whether the document will be requested in the future. However, since it does not include any aging information about the document, this cannot be used alone as the deciding factor.
- **Document Size:** The document size is another important factor for caching. In proxy caching the cached documents can be of different sizes. Having more documents in the cache will likely lead to a higher hit ratio, so one might choose to cache more small documents at the expense of performance for larger documents.
- **Type of Document:** The type of the document can also be an important factor to consider. Actually, many of the requested objects are rather small image _les, suggesting that a bias for document type could be beneficial.
- **Latency:** It is also important to consider the cost incurred in acquiring the document. The more expensive the document to download, the better it is to retain the document in the cache because the penalty for a cache miss is greater.

*B. Web caching architecture*

A web caching infrastructure is built using groups of *HTTP* proxy servers that share cached objects. To make cache cooperate on large scale and effectively increase the cache population , several caches are usually federated in caching architectures. In the internet there are a lot of caches with no communication between them these are isolated caches. The performance of a web cache system depends on the size of its client community. Bigger the user community, the hit rate increases with the number of clients connected to the caches. Thus, to increase the hit rate a simple solution is to make all these isolated caches cooperate between them.

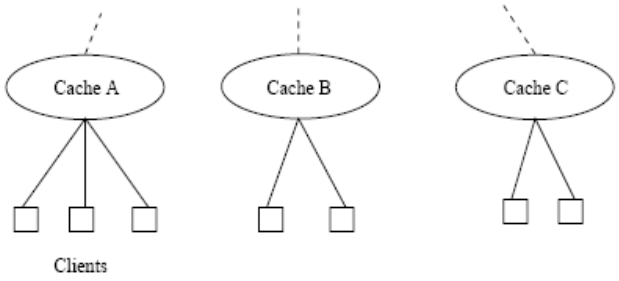



*Fig 1.3 Isolated Caches*

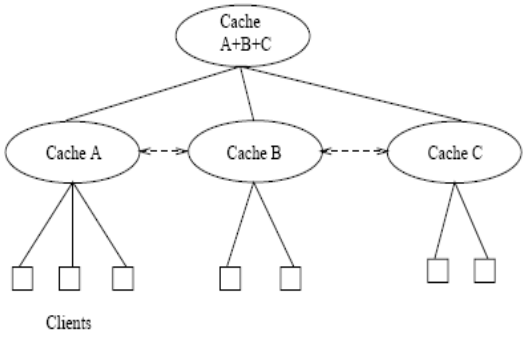



*Fig 1.4 Cooperative caches*

## II. LITERATURE REVIEW

***Joseph Kee Yin Ng et al*** [1] examined that client side caching is complementary to data scheduling in improving the performance of real time information dispatch systems.An effective caching mechanism retains data items that are most likely to be accessed by clients and reduces the number of requests submitted to the server over the wireless communication channel. This saves the narrow bandwidth and reduces the workload on the server and help reduces access latency by serving requests locally with data cached at the clients. For this purpose firstly they explored a number of scheduling algorithms i.e First In First Out (FIFO), Earlier Deadline First(EDF) for real computing, Most Request First(MRF) which is a greedy algorithm that serves the data object with the most request first, Longest Total Stretch First(LTSF), Least Recently Used(LRU), LRU-expire, Least slack first(LSF) algorithms at the client side and afterwards by using this algorithm as reference,They proposed a new broadcast algorithm called most request served(MRS) and its variants with other scheduling algorithms.

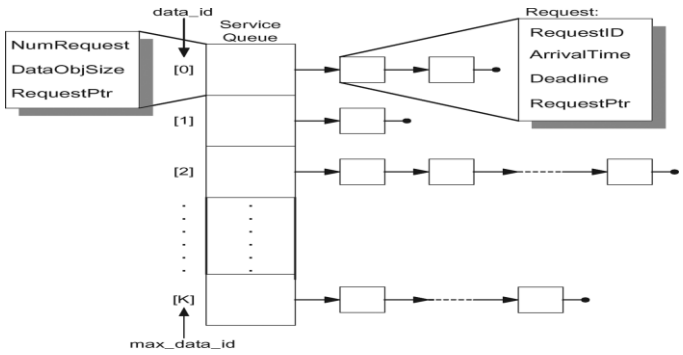



*Fig. 2.1 Common data structure used among our algorithms.*

This data structure is basically an array of K Service Queues where K is the maximum number of data objects to be served in the system. Each service queue is in turn serving a data object identified by a data_id . Each array element of the Service Queues contains three attributes keeping track of the number of requests
(NumRequest), the size of the data object (DataObjSize) , and a pointer pointing to a list of Request (RequestPtr). This list of requests refers to each distinct request for the data object under the same data_id and each node contains the following information about the unique RequestID, its arrival time (ArrivalTime), and its deadline. With this proposed algorithm performance are measured by Percentage Finished in Time =(Number of Request finished in time/Total Number of requests in the system.), Response Time, Expected Response Time= ($\sum^{M}_{i=1}$ h(i)/M ) where M is the total nimber of requests in the system and service rate .

h(i)= $Deadline_i$ if it misses its deadline $Responsetime_i$ otherwise

The service rate is determined by the bandwidth of the broadcast channel. If there is enough bandwidth, then it does not matter which algorithm is adopted and data request can be served before their deadlines. Hence if the bandwidth is not enough then the choice of algorithm make a difference. A low service rate gives a long service time for a data object.

***Hideya Ochiai & Hiroshi Esaki*** [2] states that proxy servers make replications of the value managed at an origin server, initiated by a request of user applications. With the increasing growth and network workloads over the internet object replication and cache consistency have been one of the major design issues in the recent internet applications. In proposed model, h(t) is used for cache time estimation and should be correlated with valid cache time under given a and e. The estimated cache validity time as T(a,e,h). T is strictly increasing function with regard to a and e and tells the cache validity by time. When $t_1 - t_0 \leq \mathbf{T(a_1,e_1,h(t_0))}$ then the cached $v(t_0)$ is valid. The proxy should return $v(t_0)$ to the requested user. If $\mathbf{t_1 - t_0 > T(a_1,e_1,h(t_0))}$ then cached $v(t_0)$ is invalid for the user ,It must be refreshed. After the analysis of the form of T(a,e,h), They concluded that following formula approximately provides T(a,e,h). **T(a,e,h) ≈ 0.22ae exp{3.5e(e-1)}exp{-0.00963h}** used this formula to this proposed method for estimating the validity time and the absolute subtracts of numerical values between the origin and cache servers are more appropriate for the metric of cache consistency than the life time metric.

***SungRan Cho & Wolf-Tilo Balke*** [3] analyzed that for effective caching, the main problem is predicting future queries with high accuracy and than choosing appropriate cache replacement policies to achieve a maximum of reusability of recent queries or fragments.

***Pinqian Wang , Gang liu & Zhenwen He*** [4] observed that the traditionally substitution algorithms worked with the data base that used hit frequency or access time as standard to determine which data block should be replaced in replacement operation. The criteria of these substitution algorithms are more unitary and each algorithm only has correspondingly good effect on certain type of access pattern. Based on analyzing the three dimensional spatial data that have unique properties- complex structure, various types, large amount of data, the uneven spatial distribution, varying shapes, They proposed new two cache management algorithm: Object Based Cache Pool Management Algorithm, Object Association Query Algorithm which are substitution algorithms based on hit frequency and duration. It uses the buffer pool management class to manage all the cache pools. Every object cache pool inserts ,replaces, queries itself and self maintains through a separate access management mode. The basic idea of cache dynamic adjustment is : to reduce system resources contest ,primarily memory while ensuring a high hit ratio. Object association query algorithm works as query the relative ones when dispatching objects and generate a complete spatial object model as a return value including detailed attribute information. The cache management algorithm has been implemented on cache management module of three dimensional spatial data engine. Three dimensional spatial data engine adopts multilevel caching mechanism and various pools established at all levels.

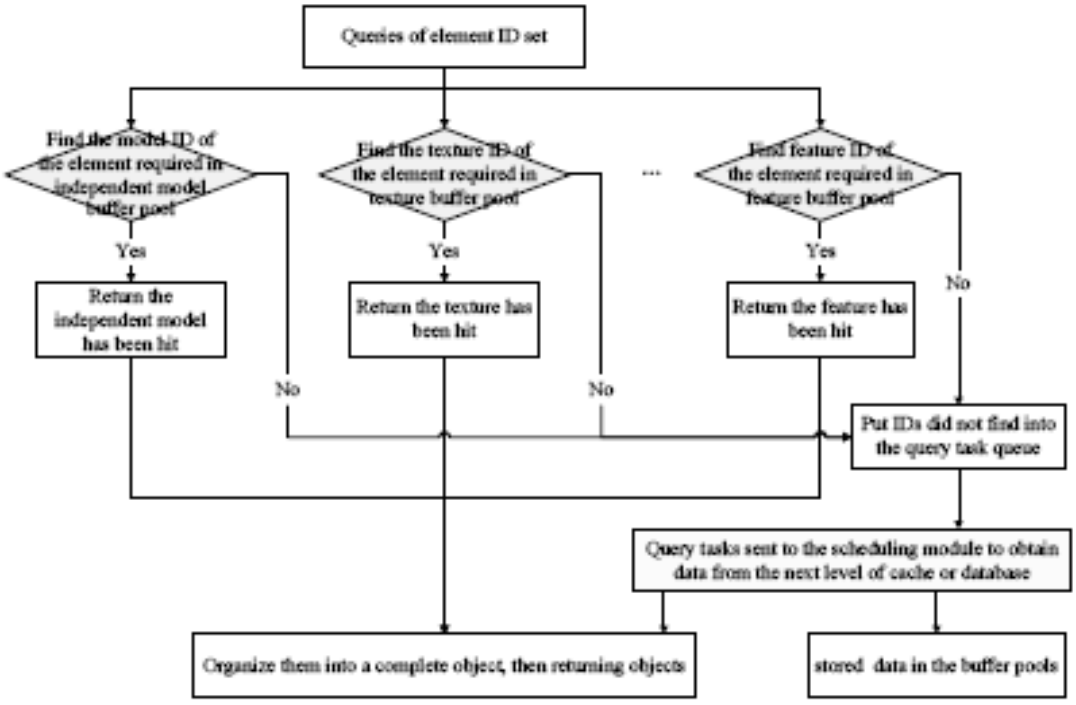



*Fig 2.2 Flow chart of object association Query.*

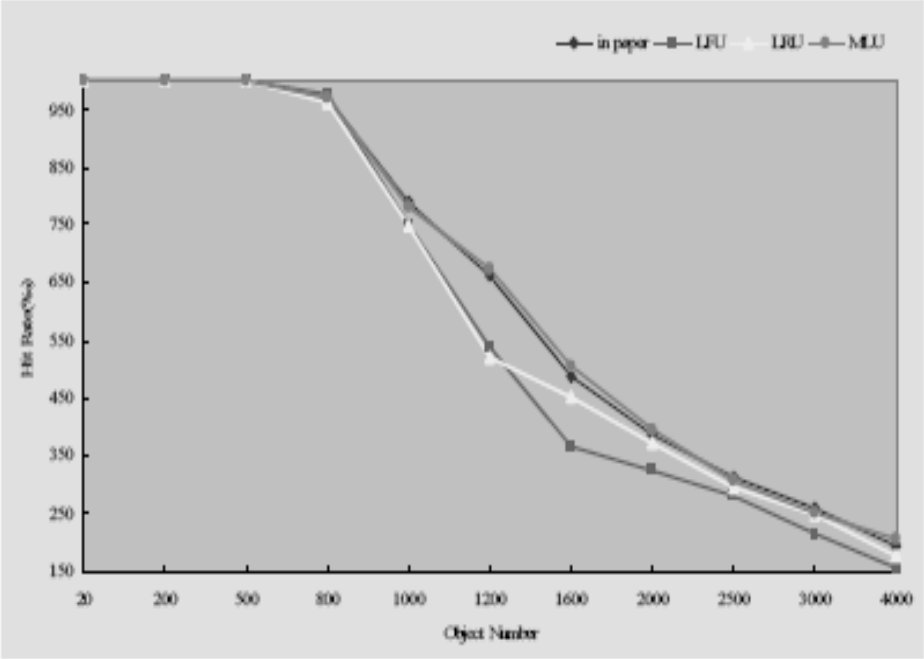



*Fig 2.3Hit rate of Replacement algorithm*

## III. PROPOSED METHODOLOGY

*A. Apache Tomcat Server*

Apache Tomcat server is one of the most popular open source web server that implements the Java Servlet and the Java Server Pages (JSP) specifications from Sun Micro system to provide the platform to run Java code on a web server provided by Apache Software Foundation. Tomcat adds tools of its own internal HTTP server and can also add tools for configuration and management. It can also be configured by editing configuration files usually written in XML format. Apache Tomcat is intended to be a collaboration of the best developers from around the world that supports only web components. Apache is developed and maintained by an open community of developers under the auspices of the Apache Software Foundation. The application is available for a wide variety of operating systems, including Unix, GNU, FreeBSD, Linux, Solaris, Novell NetWare, , Microsoft Windows, OS/2, TPF. Released under the Apache License, Apache is characterize as open-source software.

*B. Salient Feature of Apache Tomcat server*

Apache supports a variety of features, many implemented as compiled modules which extend the core functionality. These can range from server-side programming language support to authentication schemes. Some common language interfaces support Perl, Python, Tcl, and PHP. Mod Securities an open source intrusion detection and prevention engine for web applications. Apache logs can be analyzed through a web browser using free scripts such as AWStats\ W3Perl or visitors.

*C. Performance of Apache Tomcat server*

Although the main design goal of Apache is not to be the "fastest" web server, Apache does have performance comparable to other "high-performance" web servers. Instead of implementing a single architecture, Apache provides a variety of Multi Processing Modules which allow Apache to run in a process-based, hybrid process and thread or event-hybrid mode, to better match the demands of each particular infrastructure. This implies that the choice of correct MPM and the correct configuration is important.

## IV. PROPOSED ARCHITECTURE

A Proxy server is a server that can act as an intermediary for requests from clients seeking resources from other servers. A client connects to the proxy server requesting some service such as a file, connection, web page, other resources available from different server. The proxy server evaluates the request according to its filtering rules. For example, it may filter traffic by IP address. If the request is validated by the filter, the proxy provides the resource by connecting to the relevant server and requesting the service on behalf of the client. A proxy server may optionally alter the client's request or the server's response, and sometimes it may serve the request without contacting the specified server. In this case, it 'caches' responses from the remote server, and returns subsequent requests for the same content direct.

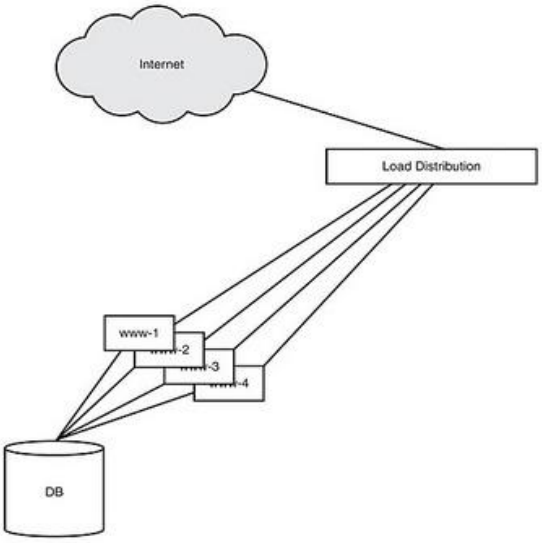



*Fig 4.1 World Wide Web load distribution*

When a browser wishes to retrieve a URL, it takes the host name component and translates that name to an IP address. A HTTP session is opened against that address, and the client requests the URL from the server. If the cache contains the referenced URL it is checked for freshness by comparing with the "Expires:" date field of the content, if it exists, or by some locally defined freshness factor. Stale objects are revalidated with the server, and if the server revalidates the content, the object is remarked as fresh. Fresh objects are delivered to the client as a cache hit. If the cache does not have a local copy of the URL, or the object is stale, this is a cache miss. In this case the cache acts as an agent for the client, opens its own session to the server named in the URL, and attempts a direct transfer to the cache.

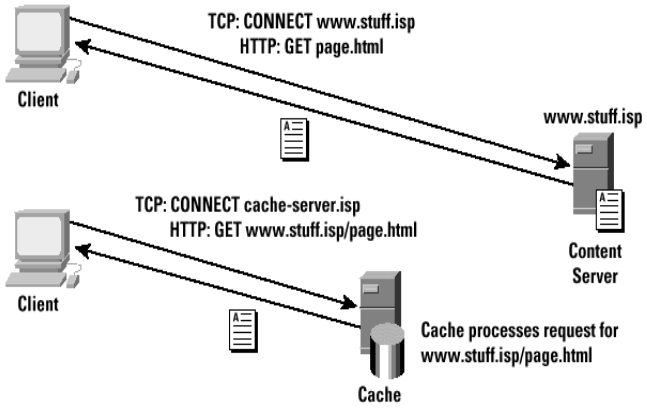



*Fig 4.2 Cache Request Processing flow*

We are going to design web caching is a Java HTTP designed/implemented for apache Tomcat (5.0 or newer). It is cache manager( that can be disabled in case of dynamic caching web sites). It's really very performant in case of static proxied web sites. Untill now, Apache Tomcat provides proxy support (http://jakarta.apache.org/tomcat/tomcat-5.0-doc/proxy-howto.html) by using Apache Web Server Web Caching support..But many times, in production applications use a servlet engine but not Apache Web Server. Moreover, many times java written applications requires java written authentication checks to identify the authorized user to access the wep resources. (typically checking session attributes). Finally, in case of static web sites it is very useful a powerful cache in order to minimize remote HTTP GET/POST for boosting performance. Thus, Fetching the static contents from the web server on to our machine and cached them reduces access latency time or saves the bandwidth next time when the user needs same resources that not need to connect directly on web server. We can get them from our caching system through the apache tomcat server installed on client machine. Apache ant is used to connect apache Tomcat server to original web server by setting the properties into pp.war file in my caching system. PP.war is a web archieve file used for web applications.

My caching system maintains caching of static contents through Apache tomcat 5.5.33 Server and apache ant. Therefore, PP. war file which is used in my caching for setting the path through its property which connects to web server where the static contents are located. Remote history fetcher class in my caching used for maintaining the history in history cache classs at the client machine because every time user extracts the static content first the object of remote history class checks the history

before connecting to original cache. Before fetching resources cache manager checks the Id to ensure the consistency whether getting update copy or old resources by checking time to live field or last modified date. My caching system then a suitable set of last positioned nodes is discarded to file system where another cache works. File System based Cache works as well. The caching (Last Recently Used) algorithm here considers a (file system) queue and for each node discarded from the in memory cache all nodes here are shifted down in the queue. If a resource is requested, the related node is pushed from the file system queue to the head of in memory queue.

## V. RESULTS

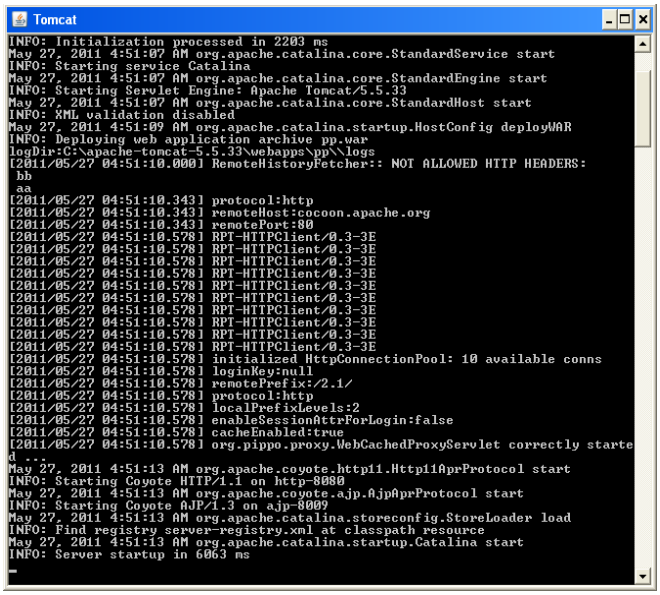



*Fig 5.1 Running Apache Tomcat web server 5.5.33 running*

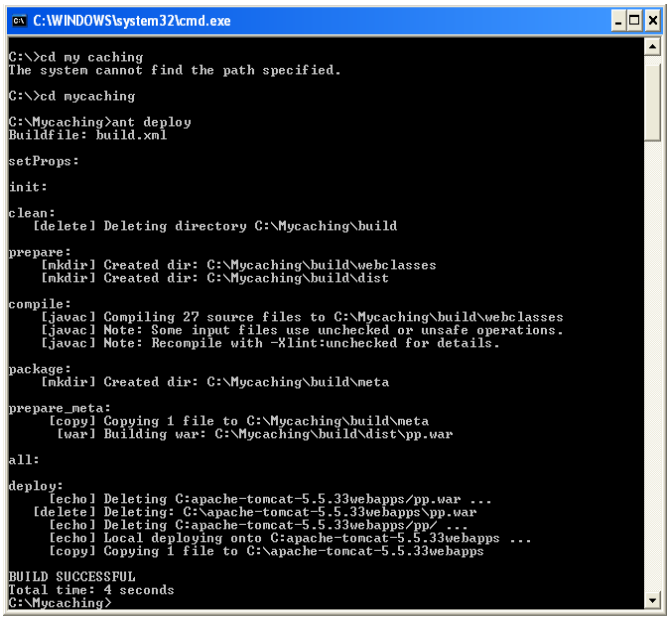
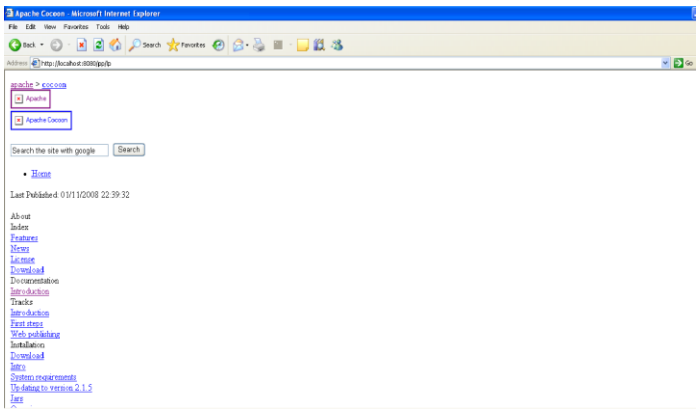
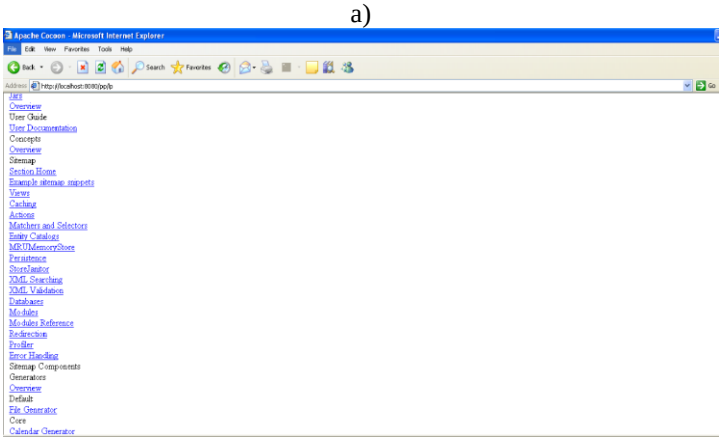



*Fig 5.2 Apache Ant has successfully deployed with Apache Tomcat server*

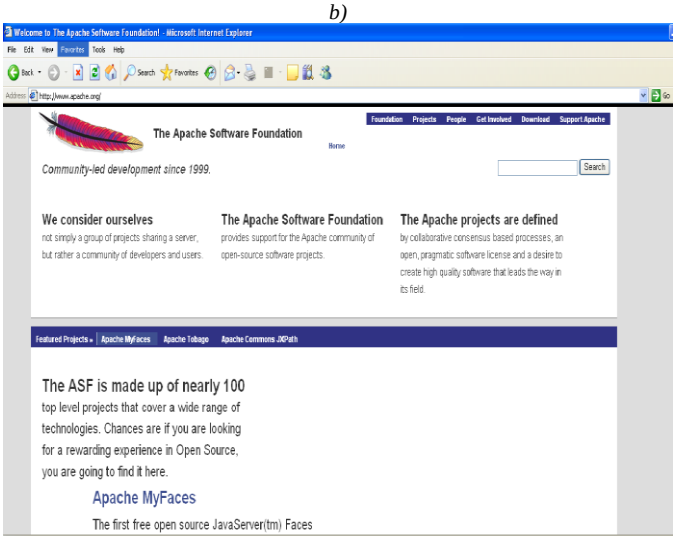
 



a)



*b)*




*c)*

*Fig 5.3 a, b, c Original Web Page of Apache.org*

## VI. CONCLUSION & FUTURE SCOPE

Web traffic has more structure to it than most type of internet traffic. Further, the characteristics of the traffic change as browsers and servers evolve, as the behavior of user changes, as the speed of the network increases and as the protocols change. A caching proxy stores the frequently-retrieved documents into local files for future use, so it will not be necessary to download the document next time it is requested. By storing popular documents closer to the users, caching proxies save network traffic and reduce Web latency. Cache response time is an important performance metric because for many users it represents the speed of the Internet. After analyzing of caching proxy response times, based on logs from real proxies located in the USA, Europe and South America. We found that high variability is an invariant in caching response times across log data of different proxies. We showed that the high variability can be explained through a combination of factors such as the high variability in file sizes and bandwidth of the client links to the caching proxies. To reduce the access latency because at a one time a number of users access the internet which introduces delays web caching is used as a fast technique.

By using web caching traffic load decreases 50% and frequency distribution made fast. We showed that simple and natural heuristics may be very effective in predicting the future traffic and estimating the value of Web documents. We suggested that a significant portion of cacheable Web traffic is static, i.e. it can be predicted for a long period of time based on the past observations. Our trace-driven simulations support this fact. We demonstrated that Static Caching enables a class of new optimization techniques. We presented in-cache compression, a potent optimization that can further reduce request response time and save network bandwidth. Finally, we discussed directions for future enhancements of Static Caching. Ideally, we could send compressed files to a client. Clearly, it does not make sense to compress documents with small compression ratio. However, about 25% of the Web traffic consists of HTML and other "ASCII" documents. These documents have an average compression ratio of 60% . An algorithm may consult a pre-computed table of average compression ratios to decide if the compression of a particular document type is beneficial.

In the future, if the portion of compressible documents remains high, then all text documents could be stored on the primary server in the compressed form. Then network bandwidth could be saved and response time decreased even without Static Caching. Our rough estimations show that this could save about 20% of the Internet bandwidth. Today, however, this would require all users to manually compress their documents which are unrealistic. Static Caching may not be applicable to all situations. The effectiveness of the Static *and other known* algorithms on servers with large portion of dynamic content is questionable and requires a further analysis. Caching of dynamic content may not be feasible at all. A possible way to deal with a frequently changing content and bursts of site popularity is to combine Static Caching with a dynamic algorithm like LRU, LFU. In this symbiosis, the Static algorithm handles the "constant" portion of the traffic, while the other algorithm keeps track on the dynamic changes in access patterns.

## REFERENCES

1. ***Jospeh Kee Yin Ng and Chui Ying Hui***.-*" Client –Side caching strategies and On demand broadcast Algorithms for Real Time Information."* IEEE , March 2008.

2. ***Hideya Ochiai and Hiroshi Esaki-".*** *Accuracy Based Cache Consistency Management for Numerial Object Replication*". IEEE Computer Society August 2008.

3. ***SungRan Cho and Wolf-Tilo Balke***.-*" Efficient Evaluation on Preference Query Processes using Twig caches"*. IEEE September 2009.

4. ***Pinqian Wang, Gang Liu and Zhenwen He***.-*" An Effective cache Management Algorithm of Three- dimensional Spatial Data Engine.*" National High Technology Research and Development Program.

5. ***Jizhong Zhao, Mix Xi and Yong Oi***. *–" Cache Predicting Algorithm based on Context aware in Pervasive Computing* " 6th IEEE /ACIS International Conference on Computer and Information Science, July 2007.

6. ***Chentao Wu, Xubin He, Qiang Ca and Changsheng Xi***. Hint-K:-*" An Efficient Multi-Level Cache using K-Step Hints"*. 2010 International Conference On Prallel Processing. IEEE, October 2010.