Dynamic Web Cache Management and Browsing Performance

Sujit Kumar Badodia
Patel College of Science and Technology,
Indore, Madhya Pradesh, India

Sachin Patel
Patel College of Science and Technology,
Indore, Madhya Pradesh, India

Rakesh Pandit
Patel College of Science and Technology,
Indore, Madhya Pradesh, India

ABSTRACT
In the daily practice we can see that the desktop applications are becomes web based. Thus the requirement for improving the web application performance is quite necessary in this day. In this paper we propose design and implement a timeline based web cache management scheme by which we improve the web applications performances. In this paper we include different aspects, problem and propose solution for the cache management strategy, additionally here we explain the working of the system at the application level. Additionally here we provide the performance evaluation of the designed system using different performance

General Terms
Cache management Algorithm.

Keywords
Web application, performance, and web cache

1. INTRODUCTION
The technology is rapidly changed and enhanced day by day, in place of the traditional computation most of the applications are becomes online or web based, to access these application at the user end various efforts and techniques are developed for increasing the performance of the system.
A web application is working with the below given scheme given in the below fig 1.

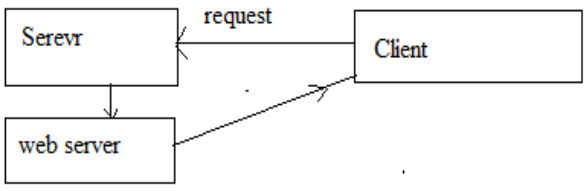

Fig 1 Simple web application working

for accessing an application from web client send a request to the server where server detect the request made by end client and search over the web server if requested page are found in the server then server respond using the requested pages. But if the application is written in any scripting language then the system required an additional application server where the requested page is executed first and then generated HTML is send to client as response. But due to large data requirement not all pages are in statically executed here required some additional database for manage the data over server for that purpose the below given system is used for application management. Suppose an application is written using (asp, aspx, php or jsp). Then there is required to execute first using an application server. And the required data is fetched from database server and serve using web pages in client machine

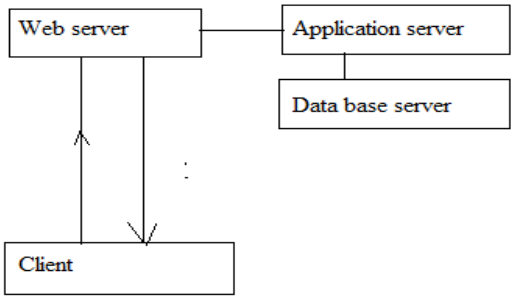

Fig 2 Database driven (Dynamic web application)

As we discussed above the application is divided in two group's first static and second dynamic. Static applications contains the static objects and dynamic pages are contains changeable objects, to manage these contains and speedup the applications client machine store frequently used data in the local system by which there are not required to fetch the data from server again in each time when requires is send to the server, it is locally provided to the browser in next time. This storage is known as the cache, which store frequently used objects from the web pages.
In this paper first analysis the different caching schemes and then proposes a most promising method for managing these objects

2. BACKGROUND
In this section we provide the different web caching schemes that are used and proposed in previous days. Additionally here we provide the desired properties by any caching scheme.

2.1 Harvest cache: Harvest cache servers [4] are organized in a hierarchy with parents and siblings and they co-operate using a cache resolution protocol called Internet Cache Protocol (ICP). When a cache server receives a request for an object that misses, it sends a request to all its siblings and parents via remote procedure call. The object will be retrieved from the site with lowest latency.

2.2 Summary cache: Summary cache [5] is a scalable wide area cache sharing protocol. Each proxy improves cache sharing by keeping a compact summary of the cache directory of every participating proxy. When a client request misses in the local cache, the proxy checks these summaries for potential hits. If a hit occurs, the proxy sends request to the relevant proxies to fetch the Web page. Otherwise, the proxy sends request directly to the Web server.

2.3 Adaptive Web caching: Adaptive Web caching [6] is an adaptive, scalable, and robust caching system. Cache servers are self-organized and form into a tight mesh of overlapping multicast groups and adapt as necessary to changing conditions. This mesh of overlapping groups forms a scalable, implicit hierarchy that is used to efficiently diffuse popular Web content to wards the demand. Adaptive Web caches exchange their entire content state with other members of their cache groups to eliminate the delay and unnecessary use of resources of explicit cache probing.

2.4 Access driven cache: Access driven Web caching [6] is a scheme using proxy profiles and information groups which are based on Web page access patterns to reduce the average number of messages among proxies for updating the cache status (comparing to Summary cache) while maintaining a high cache hit ratio. Association rule discovery is used to find and summarize the most prevalent access patterns to Web pages from trace data. Such information is then used to partition the Web pages into clusters. All proxies who frequently access some page in the same Web page cluster form an information group. When a host wants to access a Web page, it sends a request to the local proxy. If the page is not cached in the local proxy, a request is sent to the interesting site (either a proxy in the same information group or the Web server) to fetch the page, When proxy cache content changes, only proxies in the same information group are notified.

Web caching system, we would like a Web caching system to have a number of properties. They are fast access, robustness, transparency, scalability, efficiency, adaptively, stability, load balancing, ability to deal with heterogeneity,[7][2][3][4] and simplicity. We discuss them in turn.

2.11 Fast access: A desirable caching system should aim at reducing Web access latency. In particular, it should provide user a lower latency on average than that without employing a caching system.

2.12 Robustness: Robustness means availability, which is another important measurement of quality of service. Users desire to have service available whenever they want.

2.13 Transparency: A Web caching system should be transparent for the user; the only results user should notice are faster response and high availability.

2.14 Scalability: A caching scheme to scale well along the increasing size and density of network. This requires all protocols employed in the caching system to be as lightweight as possible.

2.15 Efficiency: how much overhead does the Web caching system impose on network? We would like a caching system to impose a minimal additional burden upon the network. The caching system shouldn't adopt any scheme which leads to under-utilization of critical resources in network.

2.16 Adaptivity: It's desirable [4] to make the caching system adapt to the dynamic changing of the user demand and the network environment.

2.17 Stability: The schemes [2] used in Web caching system shouldn't introduce instabilities into the network.

2.18 Load balancing: It's desirable that the caching scheme distributes the load evenly through the entire network.

2.19 Ability to deal with heterogeneity: As networks increase in scale and coverage, they span a range of hardware and software architectures. The Web caching scheme need adapt to a range of network architectures.

2.20 Simplicity: Simplicity is always an asset. [2] Simpler schemes are easier to implement and likely to be accepted as international standards. We would like an ideal Web caching mechanism to be simple to deploy.

3. PROPOSED WORK

In the previous sections we can see the expected parameters and desired properties that lead to implement a new technique for cache management.

The new system is having some key challenges that are first need of dynamic web caching for web pages but dynamic pages are just contains page layout and data base driven data which is call using the query stream or others method. Secondly the cached object management which is stored in local disk and required to clear them time to time when required. For that purpose required a new framework for managing the cache.

The proposed work is divided into three main modules:

3.1 Web page analysis: This analysis is derived for finding the URLs and static and dynamic data which data is changed due to time and which data is statically served each time. as given in the diagram (Fig 3) the web page is when opened in web browser is read by the system and the classify the data according to their source and their type in two different categories static objects and dynamic objects. In the next phase data is used to store.

3.2 Analysis of APIs: Some time web pages contain the third part open source APIs that objects are directly synchronized by our system. for example if we find that our requested page contains a stock market chart which is provided by the any web site then when a request found by the system it is perfected for use.

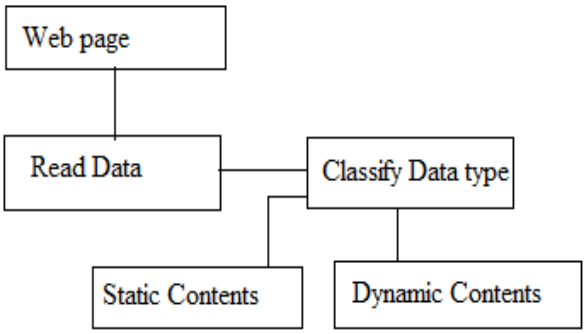

Fig 3 web page serving systems

3.2 Cache storage management: Here we adopt the below given steps for managing the storage of the cached objects.

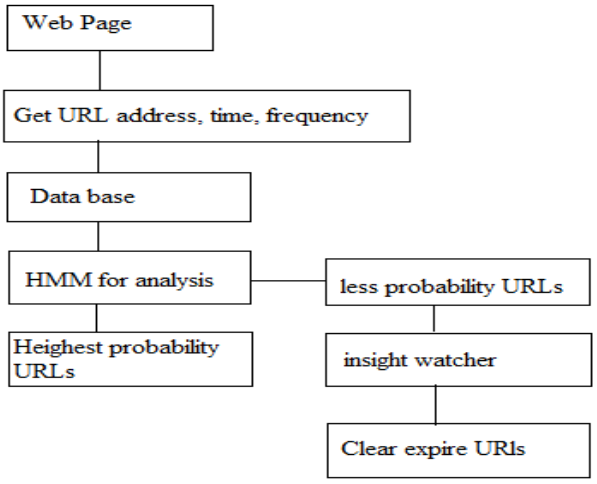

Fig 4 Shows System Proposed

as the above diagram (Fig 4) we can see that the system is working with session wise web access and their probability to manage them, for that purpose system get the URL address, Time and the frequency of opening the URLs in each session and store them over database.

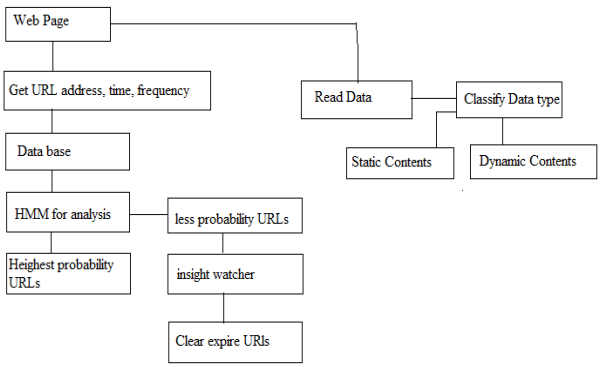

Fig 5 Shows System Architecture

The database is analyzed using the HMM model for finding the most visited URL address these URLs are not required to clear the less visited web URLs are added to the next list the list working with the time if time is expires the URL and its cached objects are cleared from the database. the complete architecture of the system is given in Fig 5.

The database is analyzed using the HMM model for finding the most visited URL address these URLs are not required to clear the less visited web URLs are added to the next list the list working with the time if time is expires the URL and its cached objects are cleared from the database. The complete architecture of the system is given in Fig 5.

4. ALGORITHMS USED

Here provides the method used to find the probability of frequently used websites and rarely used websites and the threshold to manage the desire dynamic nature of system for that purpose. Here provided the hidden markov model and Proposed cache management algorithm

4.1 Hidden Markov Model: for predicting the direction of search after personation to use as predictive model. Hidden Markov models are widely used in science, engineering and many other areas (speech recognition, optical character recognition, machine translation, bioinformatics, computer vision, finance and economics, and in social science).[5]

The Hidden Markov Model (HMM) is a variant of a *finite state machine* having a set of hidden *states*, $\boldsymbol{Q}$, an output *alphabet* (observations), $\boldsymbol{O}$, transition probabilities, $\boldsymbol{A}$, output (emission) probabilities, $\boldsymbol{B}$, and initial state probabilities, $\boldsymbol{\Pi}$. The current state is not observable. Instead, each state produces an output with a certain probability ($\boldsymbol{B}$). Usually the states, $\boldsymbol{Q}$, and outputs, $\boldsymbol{O}$, are understood, so an HMM is said to be a triple, ($\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{\Pi}$).

Hidden states $\boldsymbol{Q} = \{ q_i \}$, $i = 1, \ldots, N$.

Transition probabilities $\boldsymbol{A} = \{a_{ij} = P(q_j \text{ at } t+1 \mid q_i \text{ at } t)\}$, where $P(a \mid b)$ is the conditional probability of a given b, $t = 1, \ldots, T$ is time, and q_i in $\boldsymbol{Q}$. Informally, $\boldsymbol{A}$ is the probability that the next state is q_j given that the current state is q_i.

Observations (symbols) $\boldsymbol{O} = \{ o_k \}$, $k = 1, \ldots, M$.

Emission probabilities $\boldsymbol{B} = \{ b_{ik} = b_i(o_k) = P(o_k \mid q_i) \}$, where o_k in $\boldsymbol{O}$. Informally, $\boldsymbol{B}$ is the probability that the output is o_k given that the current state is q_i.

Initial state probabilities $\boldsymbol{\Pi} = \{p_i = P(q_i \text{ at } t = 1)\}$.

The model is characterized by the complete set of parameters: $\boldsymbol{\Lambda} = \boldsymbol{\{A, B, \Pi\}}$.

Let $\alpha_t(i)$ be the probability of the partial observation sequence $O_t = \{o(1), o(2), \ldots, o(t)\}$ to be produced by all possible state sequences that end at the i-th state.

$\alpha_t(i) = P(o(1), o(2), \ldots, o(t) \mid q(t) = q_i)$.

Then the unconditional probability of the partial observation sequence is the sum of $\alpha_t(i)$ over all N states.

The Forward Algorithm is a recursive algorithm for calculating $\alpha_t(i)$ for the observation sequence of increasing length t .First, the probabilities for the single-symbol sequence are calculated as a product of initial i-th state probability and emission probability of the given symbol $o(1)$ in the i-th state. Then the recursive formula is applied. Assume we have calculated $\alpha_t(i)$ for some t. To calculate $\alpha_{t+1}(j)$, we multiply

every $\alpha_t(i)$ by the corresponding transition probability from the *i*-th state to the *j*-th state, sum the products over all states, and then multiply the result by the emission probability of the symbol $o(t+1)$. Iterating the process, we can eventually calculate $\alpha_T(i)$, and then summing them over all states, we can obtain the required probability

4.2 Proposed Cache management algorithm:

In this subsection we provide the cache management algorithm which combination of HMM and our proposed time slice based approach for managing the cache data

1. Create cache data according to page
2. Generate last sequence of web page access
3. Calculate the probability of web page access using above HMM
4. if probability <= threshold then
5. Put into watch list
6. else
7. Put into main list
8. If watch list. date difference >=5 then
9. Clear all
10. else
11. Do nothing

Threshold values calculation: threshold values of the algorithm are defined by user access pages. Suppose a user access about 100 pages then each page are having .01 probabilities. And between only 5 pages are accessed frequently then the upper values of threshold is calculated by 1/5 = .2 thus here for experimentation we take input by user number of pages to access.

5. IMPLEMENTATION

In this section owe provide the implementation of the system .The implementation of the system is performed using visual studio dot net framework, which is combination of rich class library and an advance IDE. This IDE supports various tools that provide the programmer friendly environment for application development and deployment.

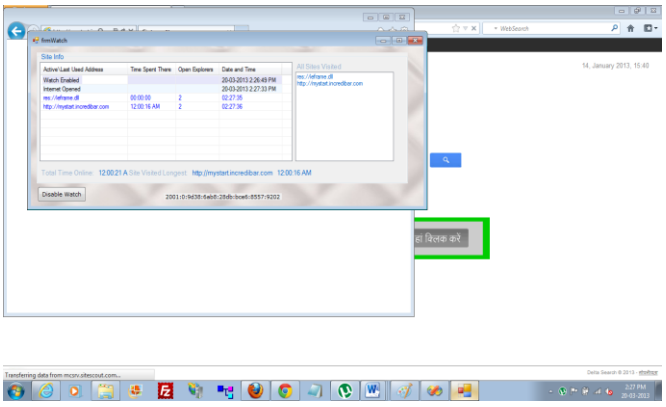

Fig 6 shows the web URL collection

The above given Fig 6 shows the collection of web URL according to the session.

6. RESULTS

The evaluated results for the different kind of utility in the application are here found using the different performance parameters. The search results accuracy is defined using the user relevance feedback.

6.1 Hit ratio: The hit ratio of the system is defined as the total number of objects is stored and utilizes the objects in the data base. The given fig 7 shows the performance of LRU and our implemented methods comparative study over different websites

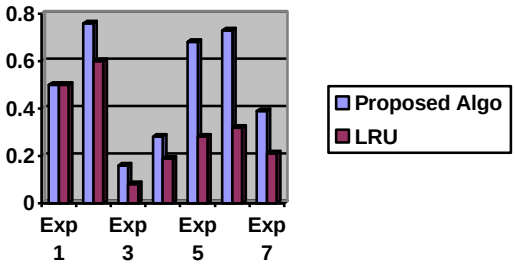

Fig 7 Shows Hit Ratio

6.2 Memory uses: This parameter is indicates the amount of memory uses for successfully execution of the system. Here the memory requirement is given in term of KB.The given Fig 8 shows the memory used by system.

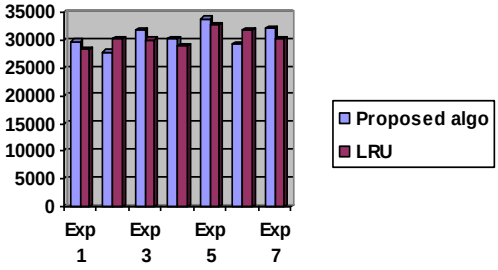

Fig 8 Shows Memory Uses

6.3 Response time: The time required to find the requested page from the system is given as response time. The response time is measured in term of seconds.

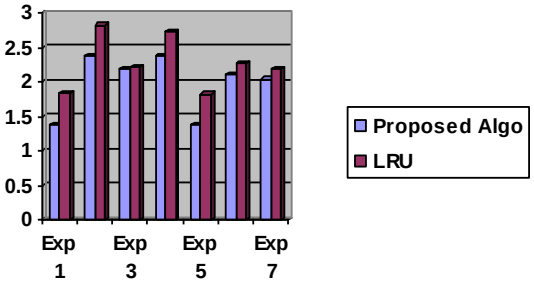

Fig 9 Shows Response time

6.4 % uses of cache: The total % of cache used during different experiments. That is evaluated using the following formula.

% uses = (number of used objects/total number of object cached)*100

Fig 10 shows the % of cache used by system.

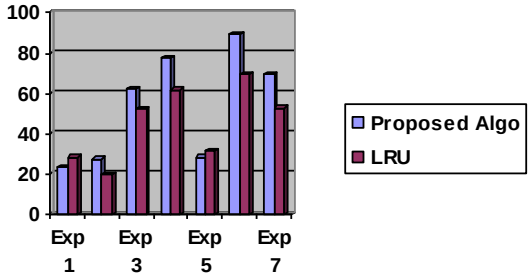

Fig 10 Shows % uses

7. CONCLUSION AND FUTURE WORK

In the proposed work we are building an intelligence model for web cache management. Which is designed using the HMM and time line based method. The proposed method is much effective for static web pages and we found that the performance of system is much higher for static contents but less effective for dynamic data contents. In future we work more about the proposed model and improve the performance for the dynamic web pages and their contents. The main issue is security to synchronize the database from the remote

8. REFERENCES

[1] An Overview Of Web Caching Replacement Algorithms, IEEE Communications Surveys & Tutorials • Second Quarter 2004

[2] Modified Pseudo LRU Replacement Algorithm, Proceedings of the 13th Annual IEEE International Symposium and Workshop on Engineering of Computer Based Systems (ECBS'06), 0-7695-2546-6/06 $20.00 © 2006 IEEE

[3] Study of Cache Management Algorithms, 1Supriya Kamoji, 2Dipali Koshti, International Journal Of Computer Science & Technology, Ijcst Vol. 2, Issue 4, Oct. - Dec. 2011

[4] A Hierarchical Internet Object Cache, Anawat Chankhunthod, Peter B. Danzig, Chuck Neerdaels, Computer Science Department, University of Southern California

[5] Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol, IEEE/ACM Transactions On Networking, Vol. 8, No. 3, June 2000

[6] URL Forwarding and Compression in Adaptive Web Caching, 0-7803-5880-5/00/$10.00 (c) 2000 IEEE

[7] Outperforming LRU with an Adaptive Replacement Cache Algorithm, 0018-9162/04/$20.00 © 2004 IEEE 4 Computer Research Feature Published by the IEEE Computer Society

[8] An Adaptive Dynamic Replacement Approach for a Multicast based Popularity Aware Prefix Cache Memory System, InterJRI Computer Science and Networking, Vol. 1, Issue 1, December 2009

[9] An Enhance Approach for Dynamic Web Caching, Sujit K Badodia,Sachin Patel, International Journal of Computer Applications (0975 – 8887) Volume 61– No.18, January 2013, ISBN : 973-93-80872-37-5

[10] Web Caching Resources www.web-cache.com

[11] Squid cache http://www.squid-cache.org/

[12] Web Prefetching: Costs, Benefits and Performance, Yingyin, Jiang, Min-You Wu, and Wei Shu,Department of Electrical and Computer Engineering, The University of New Mexico, Albuquerque, NM 87131, USA

[13] Improving Web Server Performance by Caching Dynamic Data, Arun Iyengar and Jim Challenger IBM Research Division, T. J. Watson Research Canter

[14] A Study of Bare PC Web Server Performance for Workloads with Dynamic and Static Content, Arun Iyengar and Jim Challenger IBM Research Division, T. J. Watson Research Center

[15] Web Log Mining for Improvement of Caching Performance, Rudeekorn Soonthornsutee1, Pramote Luenam Techniques For Efficiently Serving Data And Dynamic Data At Webservers Using Internet And Intranet Technology

[16] Http Cache Management To Improve Web Application Performance, Sujit Kumar Badodia, Sachin Patel and Rakesh Pandit,VSRD International Journal of Computer Science & Information Technology, Vol. 3 No. 4 April 2013,e-ISSN : 2231-2471, p-ISSN : 2319-2224 © VSRD International Journals : www.vsrdjournals.com