Energy Efficient Proxy Prefetch-Cache Framework in Clustered Architecture

1Sirshendu Sekhar Ghosh, 2Aruna Jain

Department of Information Technology, Birla Institute of Technology, Mesra Ranch-835215, Jharkhand, India
1ssghosh@bitmesra.ac.in, 2arunajain@bitmesra.ac.in

Abstract—The dynamic nature and explosive growth of the World Wide Web (WWW) makes a big challenge for retrieving and satisfying the Web clients of multiple varieties of interests with updated information and documents around the globe. Due to enormous traffic in the network and various constraints such as limited bandwidth availability, processing time at server, round trip delay etc., the Web latency is increasing continuously. A proxy based hybrid cache-prefetch model with Web log mining technique is the most attractive and successful solution to reduce this latency and to improve Web Quality of Service. Also, in today's eco-friendly computing world low power consumption and efficient energy management of individual hardware components, system software, to network applications is a critical and urgent issue. To address these issues, we propose an Energy Efficient Intelligent Agent (IA) controlled combined proxy prefetch-cache framework in clustered architecture. The proposed framework will satisfy most of the clients' requests through a local proxy cache populated from various distributed proxies within same or neighbouring clusters and very few requests are sent to the remote origin servers. It also keeps network congestion minimum by providing high hit ratio (HR) and byte hit ratio (BHR) through prefetching and minimize Web latency. We have also developed a Power-Performance model along with some guidelines for energy efficient design solutions to be adopted into our proposed framework. The performance of the proposed architecture is evaluated using hit ratio and latency time of requests depending on varying cache size and user load. The results obtained show that the proposed scheme is effective in comparison to the existing techniques used individually with respect to the above Web metrics.

Keywords— Prefetch-Cache framework, Distributed clustered architecture, Hybrid Proxy caching, Hit ratio, Response time.

I. INTRODUCTION

Web has grown rapidly in terms of number of users [1] and number of Web applications [2] from simple static information and images sharing system to sharing of highly interactive and dynamic services. Quality of Service and performance experienced by clients are critical issues in the Web domain. Many times users have to face long and unpredictable delays while accessing Web objects in real time with limited bandwidth due to heavy network congestion, bottleneck and frequent disconnections of overloaded servers. The net effect of this scenario resulting in day by day increase in the access times of Web documents i.e. Web latency or User Perceived latency (UPL). To maintain its functionality all these latencies must be maintained within the tolerable limits. A sophisticated integration of Web prefetching and Web proxy caching can significantly reduce the UPL by predicting and storing the most frequently and next future Web objects to be accessed by users into a proxy cache [3,4]. Web proxies employing combined prefetching and caching scheme with Web log mining technique reduce bandwidth consumption and underutilization, network congestion and traffic, improves reliability, can effectively serve more users’ requests reducing the heavy workload from the origin Servers and also protecting them from the “flash crowd” events. We have designed and implemented an integrated Cache-Prefetching System Model Architecture [5,8] using a novel Hybrid cache replacement policy [6] deployed at proxy server for improving Web Quality of Service towards Web latency reduction. To alleviate access of Web documents and satisfy end users’ repeated requests with minimum delay through proxy servers residing within several clusters is a challenging and relatively emerging research scope. Moreover, Eco-friendly or Green computing will be a key challenge for both information technology and business which aims at environmentally sustainable computing and responsible use of computers and related resources. Based on the report of the US Environmental Protection Agency (EPA), "the servers and data centres in USA alone consumed about 61 billion kilowatt-hours (kWh) at a cost of $4.5 billion, which was about 1.5% of the total U.S. electricity consumption in 2006, and this energy consumption is doubled in 2011 by continuously powering computer servers and data centres using the same methods [7]. Web servers at the data centres are always operating in 24X7 mode to provide uninterrupted services to the Web users’ explosive and always accessing demand. Because Web servers are at the core of data centres, so the power they consume and the heat they generate drives air conditioning costs are the prime target for energy-savings measures. Concerning clients’ request satisfaction and Web object transfer through proxies scattered into multiple clusters of distributed architecture by reducing latency as well as today’s and in future highly demanding goal to minimize the overall impact of Information and Communication Technologies (ICT) on the environment by eco-friendly performance-power aware server designing objective, in this paper we have proposed a Distributed Energy Efficient Intelligent Agent framework approach into our already
implemented integrated Cache-Prefetching System Model Architecture. We have developed an Intelligent Agent (IA) controlled combined proxy prefetch-cache framework. It is a combination of cooperative, hierarchical and distributed Web caching with prefetching and log mining techniques. We have implemented a stepwise client request handling algorithm successfully justifying the proposed combined strategy for clustered network. The rest of the paper is organized as follows. Section 2 describes the background and related work in this area. In Section 3 we introduce our proposed Novel IA controlled proxy based combined prefetch-cache model in clustered architecture along with its Power-Performance model and energy efficient design guidelines. Section 4 describes the Work functionality of the Combined Proxy Prefetch-Cache framework and phases for client’s request handling with sequence. Section 5 shows the experimental setup and results discussion. Section 6 concludes the paper showing future work towards this research direction.

II. RELATED WORK

Martin et al. [9,10] used trace-driven simulations to assess the performance of different cache replacement policies for proxy caches and utilized a trace of client requests to Web proxy in an ISP environment to assess the performance of several existing replacement policies. The results in this paper are based on the most extensive Web proxy workload characterization yet reported in the literature.

Ghosh et al. [11] presented a very detailed survey on different Web caching and prefetching policies towards Web latency reduction. They also studied various Web cache replacement policies influenced by the factors recency, frequency, size, cost of fetching object and access latency of object. They also concluded that Hit Ratio (HR), Byte Hit Ratio (BHR) and Latency Saving Ratio (LSR) are the most widely used metrics in evaluating the performance of Web Caching.

Griffioen et al. [12] paid attention to the modelling of Web prefetching and caching on file system. The research assumed that prefetching and caching share the same cache space, and showed that integrated Web prefetching and caching can improve the performance of cache system.

Cao et al. [13] presented a model of integrated Web prefetching and caching on file system, and based on which, she made performance study and simulative validation. Simulations illustrated that the integrated model could reduce the elapsed times of the applications by up to 50%.

Teng et al. [14] developed IWCP (Integration of Web Caching and prefetching) algorithm by integrating Web caching and Web prefetching which outperforms LNC-R-W3Pc algorithm in terms of delay saving ratio and hit ratio. But their algorithm was developed only for client-side proxies.

Bouras et al. [15] present an extended study about a prefetching technique and its impact on the Proxy Cache Server in a real WAN environment (i.e. university campus). The later proposal contributes with many useful considerations (e.g. log analysis, session estimation, Web object types) to take into account when prefetching is applied.

Shi et al. [16] present a model to control the prefetch requests in the Proxy Cache server side. Their mechanism tries to prevent the cache pollution caused by the prefetched objects. Therefore, if the prefetched objects replace the most popular objects cached and the cache hit ratio is decremented, the mechanism reduces the prefetch request to avoid this effect.

Wijesundara et al. [30] proposed the features in distributed Web caching concept where each client acts as a cache server and share its contents with the neighbouring node. Here Hit rate and Miss rate are considered but no discussion on cache size and latency time is done.

Nair et al. [17] presents a dynamic pre-fetching technique implemented at proxy server in which Web caching and prefetching techniques are integrated. Using the technique cache hit ratio is increased to 40%-75% and subsequently latency is reduced to 20%-63%.

Rodriguez et al. [18] analysed Web caching architectures on the basis of load, connection time latencies and transmission time latencies. Tiwari et al. [19] done a detail review on Web caching architectures like single and multiple caches, hierarchical caching, distributed caching, cooperating caching scheme, distributed Web caching with clustering, hybrid Web caching and robust Web caching schemes.

Chankhunthod et al. [19] proposed Hierarchical web caching scheme in the Harvest Project which shares the interest of a large number of clients and several countries have implemented this scheme. Here the caches are placed at the different levels of hierarchy and client’s caches are at the bottom level. For every miss the request is redirected to the next upper level caches in the hierarchy. If the document is not present in any of the level, request is forwarded to the origin server and on reply path copy of document is maintained at each intermediate level proxy server. But this scheme incurred many problems such as redundancy of data at each level and longer queries delay.

Gadde et al. [20] presented a Central Directory Approach (CRISP) to be deployed at any level of global or regional cache hierarchy in which certain numbers of caches ties together through a central mapping service to enhance capacity and scalability. Rabinovich et al. [27] altered this scheme by limiting the cooperation between the neighboring caches. This scheme has advantages by avoiding fetching of documents from the slower or distant caches if that document could be retrieved at a lower cost directly from the origin server.
Wessels et al. [21] proposed Internet Caching Protocol (ICP) and Vixie et al. [22] proposed Hyper Text Caching Protocol (HTCP), which were designed for distributed environment. They support management, discovery and retrieval of updated data from parent and neighbouring caches as well. Valloppillil et al. [23] proposed Cache Array Routing Protocol (CARP) for distributed caching where URL space is divided among an array whose elements are loosely coupled caches. Each document is hashed to a particular cache. Tewari et al. [24] proposed a scheme for fully distributed internet cache architecture, in which location hints are maintained and replicated at all local institutional caches. Rousskov et al. [25] proposed Cache Digest and Makpangou et al. [26] proposed Relais Project, in which all caches maintain local directories of contents of other caches for ease of locating documents in other caches. Also these caches keep exchanging messages with each other indicating their contents.

Lalit et al. [28] proposed the Distributed Web Caching with clustering (DWCC) approach where they provides a solution for robustness and scalability problem in web caching due to heavy load. They have used the concept of clustering along with the feature of dynamic allocation of requests by maintaining metadata of neighbouring clusters and also provide the concept of managing the load of overloaded server by transferring requests to less loaded proxy servers. Neeraj et al. [29] proposed Distributed Web Caching for Robustness, Low latency & Disconnection Handling (DWCRLD) which are based on the geographical region based clustering. They have refined their scheme to handle more delays and frequent disconnections of proxy servers. This can result in fastest response to the clients and also provide load balancing.

III. INTELLIGENT AGENT (IA) CONTROLLED COMBINED PROXY PREFETCH-CACHE FRAMEWORK

The architecture of our proposed Intelligent Agent (IA) controlled combined proxy prefetch-cache framework is depicted in Fig.1. It consists origin server (OS), proxy server clusters, IA controlled local cache (LC) and clients.

The Web is distributed around the globe and the number of proxy servers satisfying clients’ requests is grouped together into number of several clusters according as clients’ interest and behaviour as well as geographical region based. All the proxy servers (PSs) those are geographically together grouped into the same cluster. This scheme is dynamic in nature and based upon the clustering of proxy servers in distributed environment. At the highest level origin servers are scattered around the world. These OSs are connected to PSs within several clusters via various form of communication medium. The proxies employ first level of cache storing frequently referenced objects from OS and also provide cooperative document / page transfer. This scheme was referenced as Distributed Web Caching with Clustering (DWCC) technique by Tiwari et al. [28] where PS of one cluster exchange their data among themselves and whole metadata of all PSs are kept at each cluster only once for group of PSs in cluster. Periodically proxies exchange their metadata (only within the cluster) to give information of stored pages in their cache. Metadata is small and manageable so one proxy has to maintain only metadata of its own cluster only. Disadvantages associated with this scheme are: firstly as any request for any page comes on PS then it suffers from the long delays due to waiting for the metadata exchange of PSs and then it can take some action. Secondly and more importantly on disconnection of PS with other PSs it fails completely and has no mechanism to update its metadata and
act in a regular manner as was before. And lastly it has a problem of scalability as well, swamping of a PS may happen by continuous requests of clients for connection and proxies may stop responding and network may halt and service breakdown will happen. To overcome these limitations in our proposed scheme we employ a second level of caching (a local cache, LC within the cluster located between proxies and clients and controlled by some Intelligent Agents). This LC accumulates the most frequently accessed objects by clients from the upward (first) level proxies. The LC contains two important modules: Prefetch-Cache Manager (PCM) and Metadata Repository (MDR). The PCM will monitor the prefetching of Web objects from proxies either periodically or when available bandwidth or in idle time and also update the LC, provide suitable replacement policy and satisfy clients’ requests. The frequently accessed objects stored in first level caches (i.e. proxies within cluster) from OS are named “cold files” whereas more frequently accessed objects those are stored into LC from upward proxies are named “hot files”. The MDR contains information about documents (i.e. metadata) stored in LC as well as PSs within same and neighbour cluster servers (CS\textsubscript{i-1} and CS\textsubscript{i+1}). Clients’ requests first of all verified by this MDR whether it is in its own LC or neighbouring cluster’s LC and it guides any request to the corresponding PS of same cluster or in neighbouring clusters and if fails then finally the request transmitted to the highest level OS. By providing a high speed document / page transfer among hot files of neighbouring LCs we can minimize the latency time of satisfying client requests without cooperative objects transfer among PSs. In our proposed strategy the IA located within the LC will entirely monitor and provide smart mechanism to fetch objects from PSs, populate the LC with hot files, update the cache with effective cache replacement policy, search MDR for any requested object and finally keep track of high speed object transfer among LCs. We have used our Hybrid cache replacement policy [6] to import cold files or evict hot files in order to update the cache content into LC. This approach significantly reduces latency time and improves hit ratio by satisfying user requests without cooperative objects transfer among PSs, thus reduce network cost because updating metadata in most cases is quick and cheaper than updating the (potentially much larger) requested Web objects. The hot files will be migrated between LCs with extremely high throughput and low latency, since most caching systems are deployed with high-speed interconnect between cluster servers in order to meet the needs of large scale compute-intensive applications. Thus our proposed distributed IA framework approach is a combination of both cooperative and hierarchical Web caching with prefetching technique that can be easily deployed in the future Web cluster architecture. The proposed scheme will enhance scalability, alleviate extra overhead of metadata management of the PSs and also reduces the network traffic as well. To make our proposed IA controlled combined proxy prefetch-cache framework energy efficient we also propose a Power-Performance model which is a combination of proxy based Power and Performance estimation model. Among them, the Power estimation model estimates power changes and energy consumption of proxy servers working in a cluster serving users in the network in different states and loads and also shows which factors can influence the power consumption. For Performance estimation model of a proxy server, we discuss which parameters influence the response time and also estimate performance improvement of proxy server.

Fig.2 The network architecture of the client-proxy server cluster system model

Fig.2 shows the basic network architecture of the client-proxy server cluster system model containing three parts: clients, a router and the proxy server cluster. The principle of these three parts works as follows: Clients issue requests to router acting as load balancer which selects one of the proxy servers from the cluster depending upon the load and then allocates the request to a less loaded proxy server. Upon receipt of a service request from a user, a proxy server provides amount of the corresponding service with the certain energy consumption. In general, the electric power consumed by a proxy server is related with the basic running software energy consumption. In general, the electric power consumed by a proxy server is related with the basic running software energy consumption. Among them, the Power estimation model is a combination of proxy based Power and Performance estimation model of a proxy server, we discuss which parameters influence the response time and also estimate performance improvement of proxy server.

$$P_i(t) = f_i(l_i(t)) + P_i^{ACTIVE} ; l_i(t) > 0$$

$$ = P_i^{SLEEP} ; l_i(t) = 0$$

where $P_i(t)$ denotes the amount of electric power a proxy server $i$ consumes at time $t$, $f_i$ denotes the function to show the...
relation between consumed power and the load, \( l(t) \) represents the load of the server \( i \) at time \( t \) and finally \( P_i^{\text{ACTIVE}} \) and \( P_i^{\text{SLEEP}} \) are the power consumptions of the server \( i \) in ACTIVE and SLEEP state. In general, if the load is more, the larger the power consumption becomes. The response time is the most important factor in estimating the performance of a proxy server. It is expressed in terms of the round trip time (RTT) between a user and a proxy server and the current amount of load \( l(t) \) in the server. The RTT is determined by the distance and the bandwidth between a user and a server. Sometimes, the round trip time gets bigger due to the congestion of a network. Another important factor which influences the response time is amount of load in a server. That is because: if the current load is larger, each request needs to be kept in the queue of the server; hence it takes longer time for each request to be proceeded in the server. Based on the above discussion, the response time \( R(t) \) at time \( t \) can be modelled as follows:

\[
R_i(t) = g_i(l_i(t)) + \text{RTT} \tag{2}
\]

where \( g_i \) represents the function to show the relation between load \( l_i(t) \) and response time \( R_i(t) \) at time \( t \). Energy consumption can be generally defined as: Energy = AvgPower \( \times \) Time, where Energy and AvgPower are measured in Joule and Watt, respectively, and 1 Joule = 1 Watt \( \times \) 1 Second. Energy efficiency is equivalent to the ratio of performance, measured as the rate of work done, to the power used and the performance can be represented by response time or throughput of the computing system.

\[
\text{Energy Efficiency} = \frac{\text{Workdone}}{\text{Energy}} = \frac{\text{Workdone}}{\text{Power} \times \text{Time}} = \frac{\text{Performance}}{\text{Power}} \tag{3}
\]

The main approach towards energy-efficiency is efficient power management. According to equation (3), there are two ways to enhance energy efficient computing: either improving the performance with the same power, or reducing power consumption without sacrificing too much performance. For energy-efficient systems, while maximal performance for some tasks (or the whole workload) is still desirable in some cases, the systems must also ensure the energy usage is minimized. Preferably, a computing system consumes the minimum amount of energy to perform a task at the maximal performance level. The relationship between performance and energy efficiency is not mutually exclusive. A maximal performance could also be achieved by deactivating some resources or lowering certain individual performance without affecting the workload’s best possible completion time or throughput in order to optimize energy usage. To deliver effective solutions to the energy-efficiency problem, the following six considerations can be taken as the solution design guidelines.

1. **System Components Comprehensive Monitoring**: To save power consumption, we shall first investigate where the power is spent and how to optimize the power usage.

Within a computer system acting as proxy, there are generally four energy consumers, namely, processor, disk, memory and I/O devices. Achieving energy-efficiency requires improvements in the energy usage profile of every system component.

2. **Power-Manageable Hardware Components Incorporation**: Incorporating power-manageable hardware components could help improve energy-efficiency. E.g., the voltage of hardware components can be increased or decreased through dynamic voltage scaling (DVS), which is a power management technique in computer architecture, depending upon circumstances. Dynamic voltage scaling to decrease voltage is known as undervolting, and this situation can conserve power.

3. **Building Power Models for Computing Systems**: One needs to know how a computing system is constructed and how an energy-efficient system operates. It is important to construct a power model that allows the system to know how the power is consumed, and how the system can manipulate and tune that power.

4. **System Performance Understanding and Measuring**: To counter for performance with the least power consumption, computing systems must have ways to timely understand and measure system performance related to task execution under different dynamic workloads.

5. **Constructing Energy Optimizers**: The system must accommodate an energy optimizer component, which is responsible for an energy-efficient hardware configuration throughout the system operation at all times. The optimization approaches may be based on either heuristic or analytical techniques, as indicated by Brown et al. [31].

6. **Reducing Peak Power**: Barroso et al. [33] explained that current desktop and server processors can consume less than one-third of their peak power at very low activity modes, which can thus save around 70 percent of peak power. Tsirigianiis et al. [32] indicated that almost 50 percent of peak power is actually consumed at idle.

**IV. Work functionality of the combined proxy prefetch-cache framework and phases for client’s request handling**
Whenever a client made a request for some object to proxy server first of all the queue length of each cluster is checked and if it is under tolerable limit then the queue length of corresponding PSs are checked. If the client limit of PSs has not exceeded, the client’s request is served otherwise the request is forwarded to a less loaded PS. This makes efficient load balancing in the network for proper handling of all client requests. This strategy will work in following possible phases whenever client requests a proxy server (PS) of cluster n (CS_n) for some document/object/page:

1. After receiving request from the client first of all it comes to a less loaded cluster. Then the IA send it to the respective MDR to check whether the document is present in second level cache, LC (hot files) or in any of the upward first level proxy servers (cold files) and if there is a Hit, the object is replied back to the client immediately. Object reply from hot files than cold files is much faster and requires less latency.

2. In case of Miss in Step 1, the IA forward the request to the neighbouring cluster’s (CS_n, and CS_n+1) MDR. If the document is found in neighbouring LCs hot files or it is in the neighbouring clusters’ cold files, at once the document is brought from there through the high speed interconnected channel and then replied back to the client.

3. If the requested page is not found even in the neighbouring clusters, the request is again forwarded directly by the IA to the next neighbouring clusters with a factor of 2 that are clusters CS_n-2 and CS_n+2.

4. Till now the object request does not sent to the OS and thus by checking only the MDRs, requested document latency can be minimized. In our proposed architecture to enhance scalability and reliability, we also maintain an alternate backup route of requested object transfer (cold files) among proxies of different clusters in case of hardware failure, incorrect or very long cache routing and fault tolerance.

5. If the requested document is still not found, the request is finally forwarded directly by the IA to the OS. If there is a Hit at the origin server, the page is returned back to the client.

6. If the requested page is even not present at the origin server, the IA issues a “Page Not Found” message and it is flashed back to the client.

7. The IA periodically prefetch the relevant most and next frequently requested objects and populate them into the two hierarchies of caches:
   a. First level: from OS to PSs (cold files) and
   b. Second level: from PSs to LC (hot files)

These two caches are updated via Hybrid proxy cache replacement policy and the prefetching is done at idle time or after certain time interval or when available bandwidth by an accurate prediction model using Web log mining technique. The all possible phases for a client’s request are depicted in the sequence diagram shown in the Fig.3 given below.

**V. EXPERIMENTAL SETUP AND RESULT DISCUSSION**

The proxy workload to study our proposed Energy Efficient Intelligent Agent (IA) controlled combined proxy prefetch-cache framework is obtained from the running proxy server of Birla Institute of Technology (BIT), Mesra, Ranchi, Jharkhand which is extremely popular among students, faculty members and staffs of as many twenty five departments along with various administrative sections, hostels and quarters. In our experiment the proxy traces refer to the period from 13/Feb/2012:06:45:04 to 19/Feb/2012:00:00:02 of one week.

**TABLE 1**

<table>
<thead>
<tr>
<th>Workload</th>
<th>Proxy trace</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg requests / sec</td>
<td>15</td>
</tr>
<tr>
<td>Peak requests / sec</td>
<td>30</td>
</tr>
<tr>
<td>Avg requests / conn</td>
<td>45</td>
</tr>
</tbody>
</table>
The trace is composed of 9,043 nodes and 1,165,845 Web requests with average of 1,700 users per day. Table I summarizes the characteristics of the proxy trace workload. The simulations were performed at different network loads. Also Fig. 4 and Fig. 5 demonstrate the Proxy server request rate observed of a busy working day and week.

In our experimental environment we setup three clusters each grouped by three proxies from three different departments IT, CS and ECE. Also each cluster is serving three clients and the local cache is maintained within another client system. The client systems have these configurations: processor Intel® Pentium Dual Core CPU 2.0 GHz, RAM 1 GB, HDD 160 GB having OS 32-bit Windows 7 and TFT monitors. The system specification of the typical ACPI-compliant proxy server is with processor Intel® Core2Duo 3 GHz, 8 GB RAM, 500GB hard disk, and single Gigabit Ethernet card installed. The Proxy server runs Windows Server 2008 R2 OS and it is the IIS 7.5 Web server system.

![Fig.4 BIT, Mesra daily Web proxy workload](image)

![Fig.5 BIT, Mesraweekly Web proxy workload](image)

In order to evaluate our proposed combined framework, we use two performance metrics Hit Ratio (HR) and Response Time (RT) which are the most widely used metrics in evaluating the performance of cache-prefetching system. HR is defined as the percentage of requests that can be satisfied by the local cache. RT is defined as the ratio of the sum of download time of objects satisfied by the local cache over the sum of all downloading time.

Let N be the total number of requests (objects).

\[ \delta_i = 1, \text{ if the request } i \text{ is in the local cache, while } \delta_i = 0, \text{ otherwise.} \]

Mathematically, this can be expressed as follows:

\[
HR = \frac{\sum_{i=1}^{N} \delta_i}{N} \quad (i)
\]

\[
RT = \frac{\sum_{i=1}^{N} t_i \delta_i}{\sum_{i=1}^{N} t_i} \quad (ii), \text{ where } t_i \text{ is the time to download the } i^{th} \text{ referenced object from server to the cache.}
\]

We keep track of the HTTP requests made by each client computer and we find total number of hits in all 4 strategies (Cooperative, Hierarchical, Distributive and Combined) with 512 MB of proxy cache. A higher HR indicates better user’s satisfaction and defines an increased user servicing. On the other hand, a lower RT improves the network performance and reduces the user-perceived latency (i.e. bandwidth savings, low congestion etc.)
strategies i.e. cooperative, hierarchical and distributive.

In terms of minimize in RT up to 48.33% compared to other the performance in terms of increase in HR up to 69.13% and indicate that our proposed combined framework can improve strategies give similar response time. This is because up to 25% sizes whereas for RT up to 25% load level all the four hierarchical strategies, the HR is almost identical in all cache increasing user load (in %). We find that for cooperative and cache size and calculated RT in milliseconds (ms) over framework along with the proposed Power-Performance load through a less loaded cluster. Our combined proxy based model is relatively simple yet still manages to save a significant amount of energy. Also the framework can easily be adopted and deployed in dynamic Web architecture. In future, we intend to extend our approach by inclusion of more clusters and various distributed features such as optimal load balancing, cookies and encrypted file handling and dynamic updation of number of proxies residing in clusters as per the requirement.

ACKNOWLEDGMENT

We wishes to acknowledge BIT Mesra network support staffs for supplying proxy workload traces and helping us to establish experimental proxy cluster network setup.

REFERENCES


VI. CONCLUSIONS

In this paper we have presented an Energy Efficient Intelligent Agent (IA) controlled combined proxy prefetch-cache framework. The principle objective is to satisfy most of the clients’ requests through a local proxy cache populated from various distributed proxies within the same or neighboring clusters and very few requests are sent to the remote origin server. The IA provide smart mechanism to search and retrieve a requested object within the local cache of respective or neighbor clusters, monitor prefetching and caching of objects in two levels and also distribution of client load through a less loaded cluster. Our combined proxy based framework along with the proposed Power-Performance model is relatively simple yet still manages to save a significant amount of energy. Also the framework can easily be adopted and deployed in dynamic Web architecture. In future, we intend to extend our approach by inclusion of more clusters and various distributed features such as optimal load balancing, cookies and encrypted file handling and dynamic updation of number of proxies residing in clusters as per the requirement.

Fig. 6 Hit Ratio (HR) calculated for all four strategies

Fig. 7 Response Time (RT) calculated for all four strategies

Fig. 6 and 7 depict the Hit Ratio (HR) and Response Time (RT) of our proposed Energy Efficient Intelligent Agent (IA) controlled combined proxy prefetch-cache framework compared with other three strategies taken individually. With the increase in size of the cache and user load, our proposed combined framework outperforms the other three strategies computed individually for both the performance metrics. In order to have a better interpretation of our proposed framework, we have computed HR over variable local proxy cache size and calculated RT in milliseconds (ms) over increasing user load (in %). We find that for cooperative and hierarchical strategies, the HR is almost identical in all cache sizes whereas for RT up to 25% load level all the four strategies give similar response time. This is because up to 25% load level the local proxy cache is not fully populated with objects from proxies and after reaching the specified load level the local proxy cache is started updating using our Hybrid cache replacement policy. Further analysed results indicate that our proposed combined framework can improve the performance in terms of increase in HR up to 69.13% and in terms of minimize in RT up to 48.33% compared to other strategies i.e. cooperative, hierarchical and distributive.


