WEB LATENCY MINIMIZATION USING HYBRID PROXY PREFETCH-CACHE FRAMEWORK WITH POWER AND ENERGY EFFICIENCY

Sirshendu Sekhar Ghosh¹, Aruna Jain²

¹Department of Computer Applications, NIT Jamshedpur, Jharkhand-831014, India
²Department of Computer Science and Engineering, BIT Mesra, Ranchi, Jharkhand-835215, India

ABSTRACT:

The exponential growth of the World Wide Web (WWW) in terms of size, processing power and advanced software sophistication which leads to enormous traffic in the widely distributed network and several performance diminishing factors like bandwidth availability, request processing time at heavily overloaded Web servers, round trip time and Web object’s size, the Web latency is continuously increasing. The intelligent integration of two sophisticated techniques, Web prefetching and caching, deployed at proxy server level with Web log mining is the most attractive and successful solution to minimize this latency and improve Quality of Service (QoS). In addition of this latency minimization research problem objective, power consumption reduction and efficient energy management of individual hardware components, system softwares and network applications is another critical and most emergent research issue in today’s highly demanding eco-friendly computing world. In this paper, we have designed and implemented an energy efficient Hybrid Proxy Prefetch-Cache Framework towards Web Latency minimization. Using experimental Web proxy server deployment with online trace driven network setup results analysis we have shown that, our framework can successfully minimize Web latency with less power consumption and power efficiency improvement towards sustainable and greener Web usage in future.

Keywords: Web Prefetching and Caching, Web Log Mining, Performance-Power Estimation Model, Proxy Server based Energy Efficiency Module, Sleep Proxy, Optimized Load Allocation

[1] INTRODUCTION

The exponential growth of the World Wide Web (WWW) in terms of size [1] [2], processing power and advanced software sophistication has made the network prone to heavy congestion and has increased enormous load on the existing Web servers. The obvious effect of this scenario resulting in day by day increase in the access time of Web documents i.e. Web latency or User Perceived latency (UPL). A sophisticated integration of Web Prefetching and Caching, to be deployed at proxy server level can significantly reduce the UPL by predicting
and storing the next future Web objects to be accessed by users into a proxy cache. Web proxies employing combined prefetching and caching also 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. The efficiency of Web proxy caching is influenced to a significant extent by document Placement/Replacement algorithms. If a cache is full and an object needs to be stored, the policy will determine which object to be evicted to make room for the new object. In practical implementation a replacement policy usually takes place before the cache is really full. The goal of the replacement policy is to make the best use of available resources including disk space, processing speed, server load, and network bandwidth. In today’s computing world where physical along with logical or virtual cache size is not a big issue but some significant problems such as updating the large cache which involves complex computation and thus results in an increase in response time. Therefore, we must resort to an approach, which will predict the future users’ requests and retain in proxy level cache the most valuable objects thus minimizing the Web latency.

Web prefetching is another extremely useful technique focused on Web latency reduction based on predicting the next future Web objects to be accessed by users and prefetching those in idle times or when bandwidth available. So, if finally the users request it, the prefetched objects will be already available into the cache repository. This technique takes advantage of the spatial locality of Web objects and prevents bandwidth underutilization. Therefore, bottlenecks and traffic jams in the heavily congested network are bypassed and objects are transferred faster. Thus, proxy servers employing efficient prefetching technique can effectively serve more users’ requests, reducing the workload from the origin Servers. Consequently, protecting the origin Servers from the “flash crowd” events as a significant part of the Web traffic is diverted to the proxy Servers. As suggested by Marquez et al. [3], the prefetching technique has two main components: The prediction engine and the prefetch engine. The prediction engine runs a prediction algorithm to predict the next user’s request and provide these predictions as hints to the prefetch engine. The prefetch engine handles the hints and decides to prefetch them or not depending on some conditions like available bandwidth or idle time. Each engine can successfully work at any element of the Web architecture.

In present days and in future, 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. With the limited and diminishing power sources and rapid climbing of energy consumption by computing, the commitment to reduce power consumption, efficient energy usage and environmental impact of information technology resources becomes increasingly important. We have to consider green-computing technologies to save unnecessary energy usage and reduce carbon dioxide (CO<sub>2</sub>) emissions by minimizing the power consumption of electronic devices, from home appliances to numerous running Web servers at data centres world-wide. 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 expected to double by 2011 if continuously powering computer servers and data centres using the same methods [4].” Data centres use a lot of energy, nearly 3% of the electricity consumed in the United States, according to an EPA report to Congress. 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.

We have designed and implemented an integrated Prefetch-Cache Framework Architecture [5] using our novel Hybrid cache replacement policy [6] deployed at proxy server for improving Web Quality of Service towards Web latency minimization. Concerning today’s highly motivating goal to minimize the overall impact of information and communication technologies (ICT) on the environment and desperate need for eco-friendly performance-power aware server designing objective towards greener IT solutions, in this paper we have proposed an energy efficient operation scheme to be deployed at proxy server. It is a sophisticated combination of dynamic feedback control framework and sleep proxy mechanism. We have also developed an optimized load allocation algorithm with the proposed combined strategy for proxy servers’ cluster along with some guidelines for energy efficient design solutions. The rest of this paper is organized as follows: Section 2 reviews the related work and outlines the motivation and contribution of this work. Section 3 describes the working functionality of our Proxy based Integrated Prefetch-Cache Framework architecture. Section 4 explains the proxy server based performance-power estimation model. Section 5 explains the design and work functionality of proxy server based energy efficiency module along with optimized load allocation algorithm. Section 6 shows the experimental Web proxy server deployment with online trace driven network setup results analysis and discussion. Finally, the conclusion of the work presented in this paper along with our future work direction is discussed in Section 7.

[2] RELATED WORK

Kroeger et al. [7] suggested that the use of Caching can reduce up to 26% of Latency; and the use of Prefetching can improve the Web performance up to 57%. Domenech et al. [8], point out a theoretical upper bound of 97% of Latency reduction when prediction is done in a collaborative manner between proxies and Servers. According to Wenying Feng and Karan Vij [9] the Prefetching function depends upon the size of the Web page. Bigger the size of the pages less is the chance for Prefetching. The idea is to consider the cost of the Prefetching algorithm. Chen et al. [10] shows that the Prefetching increases Cache hit ratio to 30-75%. Furthermore, the combined use of Caching and Prefetching can reduce the perceived Latency up to 60%. Adaptive Web Prefetch scheme by Heung Ki Lee et al. [11] decides which Web objects are to be prefetched by considering memory status of the Web cluster system which consists of three components: Double prediction-by-partial match scheme (DPS), Adaptive rate controller (ARC) and Memory aware request distribution (MARD). DPS scheme is
introduced to obtain the relation information of objects and increase the hit rate of prefetched data, ARC scheme is used to perform an efficient management of Prefetch memory in cluster environments and finally MARD scheme to distribute Web workload to improve the efficiency in Web Prefetch. Ossa et al. [12] focused on how to implement Web Prefetching in real environments making it compatible with commercial products and standard protocols. Wenying Feng et al. [13] built a Markov tree from a training data set that contains Web page access patterns of users. Predictions for new page requests are made by searching the highest conditional probability stored in the nodes of the tree. These predicted pages are prefetched from the Server and stored in a Cache, which is managed using the LRU replacement policy. Different from the Markov tree algorithm, the Double Dependency Graph (DDG) prediction presented by Josep Domenech et al. [14] considers the characteristics of the current Web sites to improve the performance achieved by Prefetching. The idea of the DDG is based on the Dependency Graph (DG), but it differentiates two classes of dependences: those among objects of the same page, and those among objects of different pages. With this differentiation, DDG targets at predicting not only the following page but also its embedded objects. Another probability-based method is the multidimensional matrix scheme proposed by Wenying Feng and Karan Vij [15]. Instead of using the tree structure, they present a matrix structure to store sequences of probabilities. The advantages of this model include parallel and sequential search in prediction. The algorithm does not need training data and entries of the multi-dimensional matrices are dynamically updated at run time. It is shown that the Latency reduction can be dramatically increased when the search path and the number of prefetched pages are selected at the optimal. Rangarajan et al. [16] presented a Prefetching scheme using ART1 network based clustering technique to group users (hosts) and then prefetch their requests according to the prototype vector of each group. This method may cause a considerable increase in network traffic since all objects in the same cluster are fetched. Moreover, the experiment only focused on the Prefetching and did not address the interaction between Web Caching and Prefetching. Yang et al. [17] applied the clustering results using the data cube model to the problem of integrated Web Caching and Prefetching. The authors used a variant of the k-means algorithm for clustering, so the clustering requires both a prior knowledge about number of clusters and initializing cluster centroids randomly. Safronov et al. [18] presented page rank-based Prefetching mechanism for clustered Web page accesses. The pages linked to a requested page are ranked, and the rank is used to determine the pages to be prefetched. Moreover, page rank is used as a replacement mechanism, i.e. those pages with the lowest rank are replaced with new ones. The authors heuristically defined any Web directory with 200 or more files under it as a candidate cluster. A. Jain and P. Mahanti [19] discussed two integrated models, one by combining Association Rule Based Prefetching and Caching and another based on Site Topology based Prefetching and Caching in a unified framework to enhance Cache Hit Ratio and Byte Hit Ratio. They observed Association Rule Based Prefetching and Caching Model significantly enhance Cache Hit Ratio and Byte hit ratio, but they have not considered data transmission rate. Also the validation of these two models is based on Server side data only. Bhawna Nigam and Suresh Jain [20] analyzed Dynamic Nested Markov Model on three Prefetching and Caching schemes: Prefetching only, Prefetching with Caching and Prefetching from Caching for modeling the Web Log and to predict the next access Web page. The predicted Web page will be prefetched and cached in
some Cache to save user’s navigation time to provide better navigational services to the Web users. The clustering based Prefetching presented by George Pallis et al. [21] effectively integrates Caching and Prefetching. Here the clustWeb scheme for clustering inter-site Web pages is introduced. The clusters from the algorithm are obtained by partitioning the Web navigational graph using association rule Mining techniques and the connectivity among Web pages in the graph. With this clustering scheme (called clustPref), each time a user requests an object, the proxy fetches all the objects which are in the same cluster with the requested object. The network setup results indicated that the graph-based clustering approach significantly improved network performance with higher byte hit rate (BHR) than other methods. The Prediction by Partial Match model (PPM) proposed by Zhijie Ban et al. [22] attempts to capture the changing user request patterns and fit the memory. The structure is based on a non-compact suffix tree. It incrementally inserts the newest user request and deletes the oldest one. A sliding window is used to control the number of user requests. The concept of entropy is introduced and a maximum entropy principle is used to model the outgoing probability distribution of every node in the tree. Their approach predicts the user’s next request based on the nodes with low entropy. The prediction function combines lower entropy, larger prediction accuracy rate and the longest match rule to predict the user’s next request. Different from the PPM model, Yin-Fu Huang and Jhao-Min Hsu [23] developed an access sequence miner to mine popular surfing 2-sequences with their conditional probabilities from the proxy Log, and stored them in the rule table. Then, according to buffer contents and the rule table, a prediction-based buffer manager also developed here will make appropriate actions such as document Caching, document Prefetching, and even Cache/Prefetch buffer size adjusting to achieve better buffer utilization. They designed a prediction-based proxy Server used to improve hit rates of accessed documents, the architecture of which consists of three functional components such as a Log file filter, an access sequence miner, and a prediction-based buffer manager. Qiang Yang and Haining Henry Zhang [24] present that Web Logs are used to train sequential association rules to predict Web users’ browsing behavior combining a prediction model learned from Web Log data with the state-of-the-art GDSF Cache-replacement algorithm. Experiments demonstrated that the association-rule based predictive algorithm improves both the hit rate and the byte-hit rate. Baliga et al. [25] estimate that the Internet consumes 1% of the total power consumption in broad-band-enabled countries. This consumption could increase to 4% as the access rate increases. Bianzino et al. [26] determine that browsing a single web page consumes, on average, 4.7W (instantaneous power, i.e. overall energy consumption (kWh) is computing by the product of instantaneous power and browsing time), which can grow to 16W in the case of a streaming video. This is comparable to the power consumption of a single energy efficient bulb over the same time period. In 2009, Google released information about the power consumed during an average Google search [27]. They claim that 0.0003 kWh of energy is consumed per search. Considering an average of approximately 300 million searches per day, the search company consumes 90,000kWh per day, i.e. 32,850,000 kWh per year. Most of the early works related to server power management [28,29,30,31,32,33,34,35] has either focused on the processor specifically or used heuristics to address base power consumption in server clusters [36]. This motivates us to adopt a holistic strategy for the entire proxy server level power management where we exploit the system components interactions and dependencies.
between different devices that constitute a whole computing system. Dynamic Voltage/Frequency Scaling (DVFS) has been developed as a standard technique to achieve power efficiency of processors [37,38,39]. DVFS is a powerful adaptation mechanism, which adjusts power provisioning according to workload in computing systems. Horvath et al. [40] explored the benefits of dynamic voltage scaling for power management in server farms. They also try to minimize the total energy expenditure subject to soft end-to-end response time constraints. However, this work solved the problem from hardware viewpoint, and very complex and not easy to apply in the real application. Lorch et al. [41] use predictive models to estimate future load and create a power schedule. Unfortunately, the CPU currently contributes less than 50% to overall system power, thus we focus on whole system power management. A control-based DVFS policy combined with request batching has been proposed in [42], which trades off system responsiveness to power saving and adopts a feedback control framework to maintain a specified response time level. A DVFS policy is implemented in [43] on a stand-alone Apache web server, which manages tasks to meet soft real-time deadlines. Virtual machines can be used to dynamically add or remove machines in response to change in load [44]. Virtual machines take minutes to boot or migrate and introduce performance over-heads. Kansal et al. [45] focus on power consumption estimation of virtual machines running on the same physical machine. Since the performance counters can be monitored separately for each virtual machine, they attempt to segregate the power consumption. Their estimation achieves accuracy, with errors within 0.4W - 2.4W. Several recent papers [46] have used machine learning to dynamically provision virtual machines while maintaining quality of service goals. Felter et al. [47] addresses base power consumption for web servers by using a power-shifting technique that dynamically distributes and maintains power budget among components using workload sensitive polices. Contreras et al. [48] present a power estimation model for the Intel XScale® PXA255 processors. The approach exploits the insight into the internal architecture and operation of the processor. It achieves accuracy within an average of 4% error using five processor performance counters. Christensen et al. [49] was first introduced the Sleep Proxy idea. An interesting paper [50] comes in 2007 for an implemented Proxy specifically designed for Universal Plug and Play Protocol (UPnP). Bruce Nordman et al. [50] proposed solution for a Sleep Proxy which can manage protocols such as ARP, DHCP, ICMP. Agarwal et al. [51] gave another scheme similar to Sleep Proxy which offers a hardware implementation of Sleep Proxy in a so-called “gumstix”, thought as a predecessor of future NIC. The goal of incorporating eco-friendly designing and energy efficiency is to minimize the overall impact of information and communication technologies (ICT) on the environment. The global CO2 emission in 2009 was estimated at 30.398 billion metric tonnes [52]. Gartner et al. [53] estimates that ICT is responsible for 2% of the global CO2 emissions due to its power consumption. This level of consumption is equivalent to around $50 billion (considering $0.15/kWh).

[3] PROXY SERVER BASED INTEGRATED PREFETCH-CACHE FRAMEWORK ARCHITECTURE

The Integrated Prefetch-Cache Framework architecture is implemented at proxy server as it can serve wide range and variety of clients. In today's highly demanding situation,
deployment of proxy server between Web clients and Web server is essential to minimize server overload, bottlenecks and user access latency. [Figure-1] illustrates the integrated system framework architecture.

The steps shown in [Figure-1] can be explained as follows:

1. Web clients issue object requests which are sent to the Cache manager and it authenticate the users by user names and passwords by checking the stored Preference list.
2. The cache is logically partitioned into two portions, Normal cache queue (with LFU) and Prefetch cache queue (with LRU). The Cache (Replacement) manager calls the Hybrid replacement algorithm that collects information of the requested object in cache and makes the replacement decision.
3. Cache manager checks if the object is found at cache (either in Normal cache or Prefetch cache). If found considered as Hit and send it to the client in response with minimal latency, otherwise miss and sends the request to the Web server.
4. The Cache manager also prepares lists of accessed and missed URLs and sends to the Prediction engine.
5. Prediction engine stores in a Hash table both the lists and their weight information. It also runs a Prediction algorithm to predict the next user’s request and provide these predictions as hints to the Prefetch engine.
6. The Prefetch manager in the Prefetch engine handles the Hints, generates and stores Prefetching rules at Prefetching rule depository by discovering users’ Web page access patterns by reading the proxy server’s access log periodically. Thus the users’ Preference list of objects can be generated and supplied to the Cache manager.
7. Prefetch manager also decides whether to prefetch from Web server or not depending on certain conditions like the available bandwidth or the idle time and automatically starts downloading the hinted Web objects from the Web server and send as Prefetch sequence to the Prefetch cache. It also sends the updation information as response to the Prediction engine.

[4] PROXY SERVER BASED PERFORMANCE-POWER ESTIMATION MODEL

Our proposed Power-Performance model 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. This model must accomplish the combined design architecture of proxy based feedback driven control framework and sleep proxy mechanism. A server consumes energy dependently of the server state which is either SLEEP or ACTIVE. Generally, the basic power consumed by a proxy server’s ACTIVE state is much more than that in the SLEEP state. In general, the larger the current load burdens, the larger power a server consumes. Based on the above discussion the Power estimation model can be defined as follows:

\[
P_i(t) = \begin{cases} 
  f_i(l_i(t)) + P_i^{\text{ACTIVE}} & ; l_i(t) > 0 \\
  P_i^{\text{SLEEP}} & ; l_i(t) = 0 
\end{cases} \tag{1}
\]

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, \( l_i(t) \) 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 \( l_i(t) \) in the server. The round trip time 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. Here we have assumed RTT to be a constant. Another important factor which influences the response time is amount of load in a server. That is be-cause: if the current load is larger, each request needs to be kept in the queue of the server; hence each request takes longer time for each request to be proceeded in the server. Based on the above discussion, the response time \( R_i(t) \) 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.
Energy Efficiency = Workdone / Energy = Workdone / Power × Time = Performance / Power

(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 work-load’s best possible completion time or throughput in order to optimize energy usage. Brown et al. [54] treated energy efficiency as an optimization problem. To minimize the total energy, an energy efficient system must adjust the system’s hardware resources dynamically, so that only what is needed to execute tasks is made available. Rivoire et al. [55] pointed out two major complementary ways to solve the energy-efficiency problem: either building energy efficiency into the initial design of computer components and systems, or adaptively managing the power consumption of systems or groups of systems in response to changing conditions related to the workload or environment.

[5] DESIGN AND WORK FUNCTIONALITY OF PROXY SERVER BASED ENERGY EFFICIENCY MODULE

[5.1] PROXY SERVER BASED ENERGY EFFICIENCY MODULE

In this paper, we propose an energy efficient design module to be deployed at proxy server as shown in [Figure-2]. In this energy efficient module we have proposed a combined approach consisting of two interdependent techniques: Dynamic feedback driven provisioning that dynamically reconfigure the proxy server parameters by taking feedback from proxy workload logs, analysing them and also load balancing by incorporating Sleep proxy mechanism into a server cluster that optimally distributes current load among the awaked running servers.
The proxy server continuously monitors the response time of individual requests as measured by the difference between the time the request was received at the server and the time the first packet of the response was sent to the client. Our feedback driven control framework at proxy server adaptively adjusts policy parameters based on performance analysis of the server and increase energy savings when measured response times are lower (better) than the response time goal (a threshold value), or to decrease energy savings when system is not meeting the specified threshold value. In this framework, the Proxy Server Status Monitor Module periodically monitors and measures power and performance characteristics of Proxy workload based on the current server configuration. It performs the statistical analysis and feedback those results to Performance Analysis Engine. Server Status Monitor Module consists of Workload Monitor and Performance Monitor. Workload Monitor collects information about received service requests, including request numbers, request types and average service time of accessing a client, and so on. Performance Monitor collects various performance data such as utilization of CPU and Memory. Also, Server Status Monitor Module monitors if processes works properly. Performance Analysis Engine receives workload data and performance data from Server Status Monitor Module, calculates the value of load balancing under different configurations, evaluate deviation between actual and expected load value, predicts the proxy server power-performance budget for the next observation interval based on the statistical analysis as well as history-based knowledge and finally determine whether to change proxy configuration. Configuration Regulator adaptively adjusts and reconfigures Proxy server power-performance parameters towards optimal settings such that the system can meet the power budget determined by the Analysis engine.

In our proposed energy efficient strategy we have combined Sleep proxy mechanism with dynamic feedback driven control framework in a cluster of proxy servers and the associated serving clients. We assume each proxy server has two states: SLEEP and ACTIVE. In SLEEP state there is no request from users to the proxy server and in the ACTIVE state the proxy is providing service for the users. Therefore, in general, the consumed energy in SLEEP state is much less than that in ACTIVE state. Each client in the network will login in the proxy server and provide necessary information that the proxy server will need in order to maintain load balancing of the entire network. This will be done by periodically sending an INFO message that contains these data: node-id (client-id), IP address, MAC address, Operating System, state, local time and amount of data transferred. Proxy servers by listening the INFO messages from clients and store these data in tables to keep track and information about active nodes in the network. The network load will be calculated and the threshold
value must be set depending on serving number of clients and amount of data to be exchanged. In the cluster of proxy servers each server will be either in ACTIVE or SLEEP state depending upon the threshold value set for the whole network the cluster is serving and their status will be maintained in a queue. Also, each proxy server will check periodically (preferably twice the time set for periodic sending of INFO) whether the clients it was serving are active or not. Proxy servers change status from SLEEP to ACTIVE according as maintained in queue whenever the threshold value or load limit exceeds. Optionally we can configure the default time for periodic sending of INFO, IP address of server and communication port with the server. Also we can set the number of server limits at a specific time of the day or night whenever the load is almost average and the clients of the network can be optimally served with that minimum number of servers. Our proposed techniques are applicable to individual proxy server systems and complement energy management policies for proxy servers’ cluster.

[5.2] OPTIMIZED LOAD ALLOCATION ALGORITHM

Based on our proposed framework we introduce the Proxy server Optimized Load Allocation algorithm to allocate the incoming client requests to a collection (cluster) of proxy servers. The main idea of our proposed algorithm is as follows: there are two states ACTIVE and SLEEP of servers in the cluster as we mentioned before.

- Initially, all the servers are in SLEEP state.
- If there is a request from users, the router first wakes up one SLEEP state server, and then allocates the request to the server.
- After that, if a new request arrives at the router, the router allocates the request to an ACTIVE state server whose number of requests is fewer than the trade off point, i.e., \( l_i < l_i^{\text{balance}} \), where \( l_i \) is the current load and \( l_i^{\text{balance}} \) is the maximum load of each server \( i \).
- If all the ACTIVE state servers’ loads are equal to their balance point, a router wakes up a SLEEP state server and allocates such a request to the new waked-up servers. Simultaneously the server’s state changes from SLEEP state to ACTIVE state.
- Now, there are more than or equal to one server is in ACTIVE state. Thus, on receipt of a request, a router first checks the loads of every ACTIVE state server. If there is at least one ACTIVE state server whose load is lighter than the balance load, then the request is allocated to one of that ACTIVE state server.
- If the load of every ACTIVE state server is heavier than the balance point \( l_i^{\text{balance}} \), a router is regarded to wake up a new SLEEP state server and allocate the request to that newly awaked server.
- An ACTIVE state server after finishing execution of a request, if there is no request from users being executed in a server; the server changes the state to SLEEP state.
- Finally, by using this load allocation algorithm, the entire network is efficiently served by the cluster of energy-efficient proxy servers.

[6] EXPERIMENTAL NETWORK SETUP AND RESULT DISCUSSION
The Web workload to study our proposed combined energy efficient proxy designing is obtained from 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. We have performed two sets of experimental testing one with implementing our combined sleep proxy and optimal load allocation strategy and another without the strategy. We have measured the consumed energy and the network traffic (number of packets) passing to them during the Institution working hours 08:00 to 16:00 for three continuous days with the above mentioned Web proxy workload and each client is connected with the proxy server through the Institution’s proxy client application “Cyberoam Captive Portal”. The first set of testing, Test Set 1 (TS1), without combined strategy and the second set of testing, Test Set 2 (TS2) when our combined strategy is implemented on the experimental network setup. Table I shows results regarding the power consumption and total number of packets for the two Test Sets TS1 and TS2. As we can see from the Table I, Computer C1 and C6 have higher power consumption and this happens because of the CRT screen that consumes more power.

<table>
<thead>
<tr>
<th>Computer</th>
<th>TS1 (without combined strategy)</th>
<th>TS2 (with combined strategy)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power Consumption (Watts)</td>
<td>Total no. of packets</td>
<td>Power Consumption (Watts)</td>
</tr>
<tr>
<td>C1</td>
<td>133.7</td>
<td>208734</td>
</tr>
<tr>
<td>C2</td>
<td>116.7</td>
<td>53440</td>
</tr>
<tr>
<td>C3</td>
<td>114.15</td>
<td>138331</td>
</tr>
<tr>
<td>C4</td>
<td>113.46</td>
<td>172610</td>
</tr>
<tr>
<td>C5</td>
<td>117.9</td>
<td>195652</td>
</tr>
<tr>
<td>C6</td>
<td>135.1</td>
<td>288957</td>
</tr>
</tbody>
</table>

[Figure-3] shows the power consumption vs. CPU utilization with and without implementing the combined strategy while changing the CPU utilization by using variable workloads, where the horizontal axis indicates the average CPU utilization reported by the OS, and the vertical axis indicates the average power consumption measured at the proxy server.

![Power consumption vs. CPU utilization](image)

Figure: 3. Power consumption vs. CPU utilization

We have observed two important facts. First, the power consumption increases almost linearly with CPU utilization, as reported in other studies. Second, an idle server consumes up to 66% of the peak power, because even when a server is not loaded with user tasks, the power needed to run the OS and to maintain hardware peripherals, such as memory, disks,
master board, PCI slots, and fans, is not negligible. [Figure-3] also implies that if we allow user connections for login requests to a limited number of proxy servers, and keep the rest of the servers hibernating, we can achieve significant power savings. However, the consolidation of login requests results in high utilization of those servers, which may down degrade performance and user experiences. Our measure leads to the following observations: although when in the idle state a proxy server can have null performance, it has inherent power consumed by some resources to maintain the basic routines of the system. Moreover, the power consumption shows only slight variations when in the idle state and the average power consumption is approximately constant. According to our observation, the power consumption, while in the busy state, covers a wider range of value, followed by the utilization of the servers.

Table II: TS1 AND TS2 data of Performance, Average power and Power efficiency in different load levels

<table>
<thead>
<tr>
<th>Load Level</th>
<th>Without combined strategy (TS1)</th>
<th>With combined strategy (TS2)</th>
<th>Comparison in different load levels</th>
</tr>
</thead>
<tbody>
<tr>
<td>100%</td>
<td>21,959</td>
<td>324</td>
<td>68</td>
</tr>
<tr>
<td>90%</td>
<td>19,752</td>
<td>318</td>
<td>62</td>
</tr>
<tr>
<td>80%</td>
<td>17,552</td>
<td>312</td>
<td>56</td>
</tr>
<tr>
<td>70%</td>
<td>15,360</td>
<td>292</td>
<td>53</td>
</tr>
<tr>
<td>60%</td>
<td>13,163</td>
<td>277</td>
<td>48</td>
</tr>
<tr>
<td>50%</td>
<td>10,969</td>
<td>274</td>
<td>40</td>
</tr>
<tr>
<td>40%</td>
<td>8,777</td>
<td>271</td>
<td>32</td>
</tr>
<tr>
<td>30%</td>
<td>6,584</td>
<td>268</td>
<td>25</td>
</tr>
<tr>
<td>20%</td>
<td>4,390</td>
<td>264</td>
<td>17</td>
</tr>
<tr>
<td>10%</td>
<td>2,196</td>
<td>261</td>
<td>8</td>
</tr>
<tr>
<td>0%</td>
<td>0</td>
<td>256</td>
<td>0</td>
</tr>
</tbody>
</table>

Averages: 28% 18%
WEB LATENCY MINIMIZATION USING HYBRID PROXY PREFETCH-CACHE FRAMEWORK WITH POWER AND ENERGY EFFICIENCY

Figure: 4. Performance (Responses/Sec.)

Figure: 5. Average Power (Watts)

Figure: 6. Performance efficiency (Performance/Watts)

[7] CONCLUSION

This paper presents an energy efficient Hybrid Proxy Prefetch-Cache Framework towards Web Latency minimization. The principal objective is to incorporate efficient power management into our Integrated Prefetch-Cache Framework. This objective enables always active proxy servers to improve its energy efficiency under fluctuating loads, and to dynamically match both load and power consumption. The combined strategies of Dynamic feedback driven provisioning and Sleep proxy mechanism in proxy server presented in this paper can adaptively reconfigure proxy performance parameters by efficiently measuring existing client loads, do optimal client serving load allocation among the running (ACTIVE mode) proxies and also provide energy efficiency by sending proxies into SLEEP state. Moreover to evaluate the performance of the proposed scheme from the view point of energy consumption, we have implemented an experimental client-proxy network setup with 1 proxy server and 6 client systems. The result shows that the proposed scheme can successfully reduce power consumption by 28% and improve power efficiency by 18%. Our combined energy efficient strategy along with optimal load allocation algorithm is relatively simple yet still manages to save a significant amount of energy. In the future, we plan on looking at more sophisticated power and energy predictors which will allow the intelligent proxy server to
handle a greater variety of workloads. Furthermore, power-efficient workload management, simulation tool buildup and larger network deployment of our framework is the next-step work in this direction.

ACKNOWLEDGMENT

We wish to acknowledge BIT Mesra network support staffs for supplying secure proxy workload traces for research and development purpose and helping us to establish experimental proxy server cluster network setup with suitable hardware and software support.

CONFLICT OF INTEREST

The authors have declared no conflict of interest.

COMPLIANCE WITH ETHICS REQUIREMENTS

This article does not contain any studies with human or animal subjects.
WEB LATENCY MINIMIZATION USING HYBRID PROXY PREFETCH-CACHE FRAMEWORK WITH POWER AND ENERGY EFFICIENCY

REFERENCES

WEB LATENCY MINIMIZATION USING HYBRID PROXY PREFETCH-CACHE FRAMEWORK WITH POWER AND ENERGY EFFICIENCY

WEB LATENCY MINIMIZATION USING HYBRID PROXY PREFETCH-CACHE FRAMEWORK WITH POWER AND ENERGY EFFICIENCY


[52] http://co2now.org/

WEB LATENCY MINIMIZATION USING HYBRID PROXY PREFETCH-CACHE FRAMEWORK WITH POWER AND ENERGY EFFICIENCY


Author[s] brief Introduction

Sirshendu Sekhar Ghosh has submitted his Ph.D. thesis from Department of Computer Science and Engineering, Birla Institute of Technology (BIT), Mesra, Ranchi, Jharkhand, India in 2015. He has completed his M.Tech in Information Technology from Indian Institute of Engineering Science and Technology (IIEST), Shibpur, West Bengal, India in 2010. He has done MCA, M.Sc in Mathematics and B.Sc. Hons. in Mathematics from Kolkata, West Bengal. He has more than 6 years of Teaching and 4 years of Research and Teaching Assistantship experience. Currently he is working as a Faculty in the Department of Computer Applications at National Institute of Technology (NIT), Jamshedpur, Jharkhand, India. His research interests are Internet Technology, Web Traffic Optimization and Web Mining.

Dr. Aruna Jain has received her Ph.D. from Birla Institute of Technology (BIT), Mesra, Ranchi, Jharkhand, India in the year 2008. She has done M.Tech in Computer Science from BIT, Mesra and M.Sc. in Physics with specialization in Electronics from Nagpur University, India. She has published more than 50 papers in reputed National and International Journals and Conferences. She has acted and associated as resource person in various National and International conferences, co-editor and reviewer in reputed Journals. Her Research areas are Web Engineering, Computer Networks and Security, Speech Processing, Data Mining and Soft Computing. She has more than 26 years of teaching experience and 16 years of Research experience. Currently she is working as an Associate Professor in the Department of Computer Science and Engineering, BIT Mesra, Ranchi, Jharkhand, India and guiding several Ph.D. research scholars.