ENHANCED WEB CACHING USING BLOOM FILTER FOR LOCAL AREA NETWORKS

Saravanan P, Arunkumar S, Subramaniyaswamy V and Logesh R
School of Computing, SASTRA University, India

ABSTRACT

Web caching plays a very important role in improving the speed of web page access and also to reduce the bandwidth usage. In browser level caching the browser contains its caching mechanism which is isolated from any other browsers present in the Local Area Network (LAN). The browser access pages either using local cache or from the web server. The mechanism increases access time if the cache has not been available in the local system. In this proposed system, Local Area Network level caching has been introduced to access caches from other nodes in the LAN. The LAN computer nodes have been programmed with Bloom filters to access necessary caching information. The proposed system has been tested for single node-multiple requests and multiple node-multiple requests scenarios. The experimental results show that the technique is very efficient for browsing the web on the slow connections as the transfer in the LAN will be fast. The proposed system outperforms regarding access time for accessing the web page.

Key words: Local Area Network, Web Caching, Bloom Filter, Page Request


1. INTRODUCTION

Web caching plays a very important role in improving the speed of page access and also reduces the bandwidth usage of the user. The existing caching mechanisms have been designed using any one of the following two caching systems.

- **Forward position system**: The forward cache is a cache that is outside the server network like the web browser.

- **Reverse position system**: The caching mechanism is present in front of the web server to accelerate request and improve peak server load.

Most of the existing web caching systems use dedicated hardware like proxy servers, load balancers which are costly. They serve to the full extent when used on a very large network like the network of Content Delivery Systems. Caching mechanisms deployed by the Internet
Enhanced Web Caching Using Bloom Filter for Local Area Networks

Service Providers (ISP) or any big enterprise network where the number of users is huge. The existing mechanisms can be used to their full potential only if the numbers of users are huge.

A limited number of techniques use system’s memory as web cache to store the metadata about a web cache. The meta-cache act as a lookup system for all other nodes in the network. But these techniques have complicated by peer to peer protocols which might have a single point of failures. Essentially there is no simple caching solution for a small network and a very limited internet bandwidth.

The proposed work has been aimed to develop a caching mechanism which uses the caches of the other browsers of the Local Area Network along with its cache. It has been aimed to solve the difficulty faced by the users in the slow network which are apparently trying to access the same set of resources from the server. The work has been accomplished by sharing the summary cache of the browsers in the Local Area Network among the browsers. The sharing process uses Bloom filters to reduce the data traffic among the browsers to share their cache.

2. RELATED WORK
Different types of caching are proposed using several methods to make the process of caching more efficient. Use of a catalog and meta data caching [1] was too narrow down the area of search on the internet by analyzing the type of queries generated from a particular area and then using this data to form a hash which gives a narrowed area in the world wide web to look for the data.

One way of caching mechanisms to be used on the small network was cooperative caching. For this purpose, the peer to peer protocol [2] was used for sharing the cache among other nodes. But the major drawback of that methodology was the presence of a single point of failures. These are nodes which stored the metadata of the cache in the system and acted as the lookup for other nodes the network [3, 12]. The traffic has been reduced by maintaining a cache in server location [4, 13]. This methodology also improves the load balancing. But, as the cache has been implemented in the server the average time taken for LAN nodes to load the cache from the server was high.

A breakthrough in reducing the inter proxy traffic for using the cache of the other proxies was the use of bloom filter, before this the proxy servers had to send the actual cache to the other proxy servers to make them aware of its cache. The bloom filters drastically reduced the data needed for this purpose as only the summary of the cache was being sent to other proxy servers instead of the actual cache itself. The proxy server requested the actual cache from the proxy server which contains that data only when the data is requested by some node connected to the proxy server having the summary cache [5, 6, 14].

One limitation of the Bloom filters is that the probability of false percentage increases as the bloom filter is populated. Scalable Bloom Filters [7] is the solution to this problem as the size of bloom filter has been scaled with the population.

3. PROPOSED METHODOLOGY
The proposed system has been implemented with two main concepts, these are

1. Implementation of Bloom Filter
2. Data sharing in Local Area Network

3.1. Implementation of Bloom Filter
Bloom filter is a space efficient probabilistic data structure which is used to check whether an element is a member of a set or not [8]. The false positive match is possible, but the false
negative is not positive. It can give a result implying that an element is a part of the set when it is not, but it never gives a result implying that element is not a part of the set when it is part of the set. Bloom filter has been used in many domains for filtering a specific element and checking the available features [9, 10].

An empty bloom filter is an array of m bits all set to Zero. There must also be k different hash functions defined, each of which maps or hashes some set element to one of the m array positions with a uniform random distribution. It has been fed to each of the k hash functions to get k array positions. The bits have been set at all these positions to 1.

To query for an element or to test an element for availability, the element has been fed to each of the k hash functions to get k array positions [11]. If any of the bits at these positions is 0, the element is not in the set – if it were, then all the bits would have been set to 1 when it was inserted. If all are 1, then either the element is in the set, or the bits have by chance been set to 1 during the insertion of other elements, resulting in a false positive.

Figure 1 Bit Array to test an element for availability

Figure 1 shows the Bloom filter for the set \{x, y, z\}. The colored arrows represent the mapping between the positions of the bit array and the element. The element \(w\) is not in the set \{x, y, z\} because it hashes to one bit-array position containing 0. For this figure, \(m = 18\) and \(k = 3\).

The proposed system has been implemented with the above Bloom filter shares the cache summary in the Local Area Network. It has essentially reduced the data required for synchronizing the cache in the Local Area Network.

### 3.2. Data sharing in LAN

The proposed work uses Figure 2, which explains the flow of data in the network implementing the proposed model of web caching.

Figure 2 Data Sharing in LAN Connection
Suppose, the System S3 requested a page P1 from the web, then it is cached by System S3. This information has been propagated in the LAN with the help of the bloom filter. Later, the same page P1 has been requested by the System S6 which knows that the page is already present with System S3 thus system S6 request page P1 from System S3 instead of requesting it from the web. Here in this model of cache sharing the data required to synchronize the cache is reduced drastically due to the use of bloom filters. A particular entry can be sent to the other browsers in as low as 8 bits.

Figure 3 depicts the architecture and the data flow from and to the system in the Local Area Network in detail. The proposed system contains the following four modules which have been executed as four different threads along with the main method which controls all these threads. The four modules are as follows:

1) **Initializer module**: This module is responsible for sending the current summary of the cache (Bloom Filter) to the system which just came online. This procedure has been executed only once per system. It checks for the systems continuously that which came online just now, and the system has not provided its bloom filter to them if that’s the case it sends its bloom filter and mark that it have sent its bloom filter so that it does not send it again. This module uses the address book dictionary for this purpose.

2) **Listener module**: This module is responsible for accepting the Bloom Filter send by the other system in the Local Area Network, and it also services the update request of the other system for its Bloom Filter. The module continuously listens for the incoming bloom filters, page requests and update requests. If it receives a new bloom filter, it stores that as the ip:bloom pair. If it receives an update request, then it sends its bloom filter to the requesting system. If it receives page requests, first it checks the page URL in its Bloom filter for the presence. If the page information is available then, it spawns a thread which sends that the file is available and then sends the actual file to the requesting file. Otherwise, it sends no message to the sender.

3) **Maintainer module**: This module is responsible for maintaining the cache summary (Bloom Filter) from the system it was received. This thread is spawned for every Bloom Filter received by the Listener. Periodically, It contacts the system containing the received bloom filter to get the updated version of the bloom filter.

4) **File transfer module**: This module is responsible for exchanging the files between the system which have the file and the system which needs the file.

![Figure 3 System Architecture](image-url)
The main module is acting as the interface between the user and system. It is used to enter the URL and get the web page either from the internet or the caching system. It is also responsible for starting Initializer module and Listener module at the startup. When the program has started, the listener and the initialize have been spawned by the main module. Then it acts as the interface for the user. It takes the URL of the web page requested by the user, checks in its cache for existence using bloom filter. The requested web page is served if the page exists then it exists. Otherwise, it checks all the bloom filters received by the system for the requested page. It also adds the URL in the bloom filter so that next time whenever any system requests the bloom filter the new bloom filter of summary cache is provided. This way the caches summaries have been propagated in the Local Area Network.

In the system, Cache is a temporary storage which stores the web pages that are accessed by the software from the internet. Each system contains a Bloom filter to represent the presence of the actual page in its cache memory. Each URL has been fetched from the internet and has been added to the Bloom filter.

The different bloom filters received by the program have been stored in the dictionary as IP and Bloom Filter key value pairs. IP is the address of the system from which the Bloom Filter has been received. This dictionary is used by the main module to check if there exists a system which contains a required page.

The system implements the following steps to accomplish the goal. As soon as the program starts it looks for the other systems which are online. Then the latest cache summary in bloom filter format has been sent to the system. The cache summary cache has been updated periodically by the system. The program also listens continuously on a port for receiving the bloom filter from other systems and service the update requests of the other systems i.e. it sends the latest version of its bloom filter to the system requesting an update.

Whenever an URL is entered the first time, it has been checked in the browser's cache to find out if its presence. If the resource is found in the browser's cache then it is shown to the user, else the URL is searched in the cache summary of the other received from other browsers if there is a hit then the actual resource is requested from the respective system else the URL is fetched from the internet.

This results in the reduction in the number of requests to the internet as some of the requests are fulfilled by the Local Area Network itself. This technique also reduces the average time required to fetch the page for the user giving the sense faster network to the user. The program also takes care of the dynamically changing the number of clients in the Local Area Network.

4. EXPERIMENTAL SETUP AND RESULTS

The proposed system has been implemented in LAN setup with nine browsing nodes created in Virtual LAN. Python scripts are embedded in the browsing nodes and tested with high-speed internet connections. The download speed of the internet has been 3.98 Mbps, and upload speed has been 2.44 Mbps. Table 1 and Figure 4 show the performance of the proposed system regarding multiple requests vs. time taken to load the requested page.

<table>
<thead>
<tr>
<th>Number of requests</th>
<th>Average Time taken to load the Page (Seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Existing system (Minimum Advantage)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>78.1</td>
</tr>
</tbody>
</table>

http://www.iaeme.com/IJMET/index.asp 215 editor@iaeme.com
Enhanced Web Caching Using Bloom Filter for Local Area Networks

<table>
<thead>
<tr>
<th>200</th>
<th>156.2</th>
<th>46.38</th>
<th>73.8</th>
</tr>
</thead>
<tbody>
<tr>
<td>300</td>
<td>234.5</td>
<td>68.67</td>
<td>117.26216</td>
</tr>
<tr>
<td>400</td>
<td>312.4</td>
<td>91.561</td>
<td>161.50018</td>
</tr>
</tbody>
</table>

**Figure 4** Performance of the proposed system (request vs response time)

The web page ‘www.geeksforgeeks.org’ has been requested to test the system. 0, 100, 200, 300 and 400 number requests have been requested using existing normal caching (Minimum advantage) and proposed bloom filter based web caching (Maximum advantage). Time taken for the specific request has also been given.

The time taken by the experimental setup implementing the proposed methodology takes only 0.04487 seconds in comparison to the time taken by the average browser to fetch the page from the internet which is 0.7815 seconds. It shows that the proposed methodology is better than the existing systems.

5. CONCLUSION

In this proposed system a cooperative web caching mechanism has been implemented using bloom filter for sharing the cache summary among the system in the Local Area Network. The system has resulted in the drastic reduction in the traffic in LAN for the syncing the cache. This methodology uses the cache of the other systems in the LAN along with its cache. This technique has increased the probability of detection of web pages in the LAN itself. The proposed system also eliminates the presence of any single point of failure in the network. It can act as a base for improved methodology for the cooperative web caching scenario. The proposed methodology effectively reduce the average page load time in comparison of the current systems and also give the sense of faster internet to the user by using a cache of the other systems in the network. In future, the system can be further improved by addressing the expiration of cache and the issues in dynamic web pages.

ACKNOWLEDGEMENT

Authors thank Science and Engineering Research Board (SERB) for the financial support (YSS/2014/000718).

REFERENCES


