Content-based Hierarchical Web Cache

M. Asif Iqbal, Sharifullah Khan
Institute of Information Technology,
National University of Sciences & Technology, Rawalpindi, Pakistan
{52asif, dsharif}@niit.edu.pk

Abstract - Web caches are deployed in order to quickly deliver the required information to users by placing data in proximity to users. Hierarchical Web caches are scalable, fault tolerant and enhance the degree of sharing. Typically data is either cached at each level or the most wanted data is cached at the leaf levels in the hierarchy. This strategy consumes much network bandwidth during the refreshment of data at each level and leads to poor hit ratio at upper levels by caching most popular data at leaf levels. In this paper we have proposed a content-based hierarchical Web cache. It characterizes hierarchy levels with respect to Web page contents. The highest, middle and leaf levels contain dynamic, periodic and static contents respectively. A template is maintained at each level for assembling the page contents. This proposed strategy consumes less network bandwidth in refreshment and leads to moderate processing at each level. The performance evaluation also supports the hypothesis.

1. INTRODUCTION

The Internet has become a large source of the information dissemination for systems such as e-commerce, education. With the passage of time there is remarkable increase in information growth over the Internet. It has been evolved from static Web pages to dynamic pages, such as e-commerce sites that are mainly composed of dynamic information. This made the users of the Internet in abundance. The Internet has been overloaded due to a large number of users. Consequently users face delays and failure in data delivery.

A Web cache is a technique to cache Web pages at some middleware close to end users and serves users locally. Thus, it reduces overload on original data servers, latency being faced by users and bandwidth consumption [1, 2, 7]. There are many different types of Web caches. The most common one is browser cache. It reserves small amount of space for copying data at the client’s machine. It fetches information from local machine instead of original data source. Similarly, there are proxy caches that reside between main server and clients on middleware. The other types are transparent proxy cache, reverse proxy cache.

Caching nodes/levels are more than one and arranged like a tree structure in hierarchical Web cache [2, 6, 7]. A user sends a request to a leaf node and if it finds the requested information at the leaf node then the user will be served from the leaf node, otherwise the request will be forward to the next upper level or to the original data server. Hierarchical Web cache improves scalability, fault tolerance and enhances the degree of information sharing on the Web [3, 4]. In this research, hierarchical Web caching architecture has been adopted and further optimized its processing, particularly in terms of bandwidth and time consumption.

The rest of this paper is organized as follows. In section 2, we describe the strategies of copying information at different levels in hierarchical Web cache. In section 3, we describe related work and identify their limitations. The research problems have been described in section 4. Section 5 describes the proposed framework for hierarchical Web caching. In section 6, the proposed architecture and its implementation are described. The evaluation of the system has been presented in section 7. Section 8 concludes the paper and presents the future research directions.

2. CONTENTS CACHING IN HIERARCHICAL WEB CACHING

Data are copied in more than one level in hierarchical cache. Strategies for copying the data in the cache are classified as follows

1) Leave Copy Everywhere (LCE) is a strategy when a hit occurs at a level; a copy of the requested data is cached in all the intermediate levels on the path from the location of the hit down to the requested client [2].

2) Leave Copy Down (LCD) is strategy where a new copy of the requested is cached only at the level that resides immediately below the location of the hit on the path to the requesting client [5].

3) Move Copy Down (MCD) copies data similar to LCD. However, data is deleted from the level where it was previously resided except from the original data server [5].

3. RELATED WORK

Considerable research on hierarchical Web caching has been done in the past few years. Hierarchical Web caching was first proposed in the Harvest project [6]. The paper shows that Harvest cache achieves better performance over other proxy caches. It helps in load distribution of server, reduces latency and protect network from erroneous clients. Reference [7] proposed the Volume Leases algorithm that provides strong consistency in hierarchical cache. They provide growing and shrinking of hierarchies through split and join...
mechanism. So this helps in fault tolerance and scalability issues. In [5], they compared the performance of various contents caching strategies and proposed a load balancing mechanism for hierarchical caches. Moreover, Web workload characterization has also been extensively studied in the recent past from the perspective of proxies [8, 9, 10], client browsers [11] and servers [12, 13]. References [14, 15] have designed a markup language: Edge Side Include (ESI) to accelerate the performance of dynamic web-based application by defining cacheable and non-cacheable web page components that can be aggregated, assembled and deliver at the network edge. Several commercial caching and edge server products, most notably IBM’s WebSphere [16] and Akamai’s EdgeSuite [17] use the above markup language in quasi-static template to distinguish cacheable and/or non-cacheable objects.

[18] proposed a methodology for evaluating characteristics of dynamic web contents. The methodology is used to obtain models for various independent and derived metrics such as object size, freshness time and content reusability. DYCE - their emulating tool uses this methodology. In [19], proposed a methodology – MONARCH for characterizing a Web page contents with respect to time, in static, periodic and born-on-access categories. They emphasize on strong cache consistency.

4. PROBLEM DESCRIPTION

The existing architectures of hierarchical Web caching replicate complete contents of a Web page in a cache. Moreover, the cached contents move to the leaf nodes using the existing strategies of copying information in hierarchical Web caching. Consequently leaf nodes are overloaded. The architectures adopt either strong consistency or weak consistency for updating cached contents. The architectures: (1) consume considerable bandwidth in caching and refreshing the page contents on various nodes/levels, (2) lead to greater processing at lower nodes and poor hit ratio at upper nodes, (3) require much storage at each node and (4) utilize much time in updating cached contents at each node. On the other side, the current frameworks of Web workload characterizations are not employed in hierarchical Web caching architecture. The issues to handle in this research are to: (1) reduce bandwidth consumption, (2) optimize the time of refreshing, (3) improving load on leaf nodes and (4) tradeoff between strong and weak consistency.

5. PROPOSED FRAMEWORK OF HIERARCHICAL WEB CACHING

The proposed framework, to handle the above research issues, comprises of the following policies: (1) contents categorization, (2) contents distribution, (3) contents caching strategy and (4) hybrid consistency. In the subsequent subsections the above policies are explained.

5.1. Contents Categorization

The World Wide Web has evolved beyond simply displaying static web pages. A typical Web page currently consists of contents of various characteristic. These contents are categorized with respect to their refreshment time in static, periodic and dynamic contents in this research. Static contents are those contents of a Web page that do not change for long time period, such as logos, images and profiles. Periodic contents changes after some predetermined time, such as periodic updates of news and weather forecast. Dynamic contents changes irregularly such as breaking news, stock values, The contents of a Web page are classified by author of the page using additional attributes within the existing HTML tags such as TTL attribute for periodic contents. Caching of database-driven dynamic contents is currently not considered in this paper and is included in future work.

5.2. Contents Distribution

We characterize the contents of a Web page in three different categories as described in the above subsection. We propose the distribution of different contents’ categories over different hierarchy levels. Dynamic contents are cached at the top hierarchy level. As dynamic contents change more rapidly and randomly and they contribute much to the bandwidth consumption, so by keeping them at the top much bandwidth can be saved. There will be no need to redistribute contents at lower levels. This reduces the original server load in disseminating information and the server will also keep track of only the top level cache nodes.

Periodic contents do not change as rapidly as compared to dynamic contents, so these contents are cached at middle level. The contents change after pre-determined time, so it is the responsibility of cache to update data. Static contents are cached at leaf levels. These contents remain constant and do not change for long time. Consequently, it reduces the network consumption.

5.3 Contents Caching Strategy

The contents are categorized and distributed in the proposed framework of hierarchical Web caching as described in the above subsections. In the proposed framework a template is generated by parsing the page based on additional attributes included in the page by the author. The template of a cached page is maintained at each level of the hierarchy, so a request can be served at any level.
When a request is received at node/level and the template of the requested information is not already cached at the level, then the request is forwarded to the original server. The server sends the required page in response to the request. The top level caches store dynamic contents with the page template and forward the page to the next lower level caches. The middle level caches store periodic contents with page template and forward the page to the leaf level caches. The leaf level caches store static contents with page template and forwarded to the user. When a request is received at a level and the template of the requested information is already cached at the level, then the remaining contents will be fetched from the other levels; assembled and forwarded to the user.

5.4 Hybrid Consistency

In the current approaches, either weak or strong cache consistency protocols [1] are used depending on user requirements. Both protocols cannot be used simultaneously because the complete page is cached at each level of the hierarchy without contents categorization and distribution. In the proposed framework, both consistency protocols: weak and strong, are adopted, unlike the current approaches. The strong and weak consistencies are maintained through invalidation and Time-to-Live protocols for dynamic and periodic contents respectively.

5.5 Significances of the Proposed Framework

The significances of the proposed framework due to the characterization and distribution of contents at different levels of the hierarchy are described as follows:
1) It avoids the complete replication of a page contents at each level of the hierarchy.
2) It supports load balancing through equal hit ratio at each level of the hierarchy instead of only hitting at leaf node.
3) It helps in fault tolerance and scalability issues.
4) It reduces latency of searching a page in the hierarchy due to the availability of the page template at each level.
5) It reduces contents staleness and server load in maintaining strong consistency for refreshing dynamic contents only at the top level.
6) It consumes less bandwidth in refreshing page contents.
7) It reduces the time consumed in refreshing the page contents.

However, the proposed framework may consume more time in retrieving and assembling a Web page from the various levels in hierarchy in comparison to the existing frameworks.

6. THE PROPOSED ARCHITECTURE AND IMPLEMENTATION

The proposed architecture of the framework is shown in Figure 1. The various levels of the hierarchy can be either located at one place or distributed places, such as, the dynamic level could be at county level, periodic at state level and static at city level. The architecture has been implemented using Java, Java Server Pages (JSP) and Jakarta Tomcat 5.0 Web server [20]. The server and the all caches are located at different systems connected with 100 Mb line.

The template is a page skeleton using frames, iframes and layers in HTML and used by a JSP assembler for assembling the page from various levels of the hierarchy. The assembler is installed at each level. It replaces the original URL of the page content (static, periodic or dynamic) by the local addresses of the cache. Moreover index tables (shown in Table 1) are used by JSP assembler for refreshing periodic and static contents at middle and leaf levels respectively. While dynamic contents are pushed by original data server using Invalidation protocol.

7. PERFORMANCE EVALUATION

In this section, we evaluate the performance of proposed framework in terms of bandwidth, time consumption in refreshing the page contents at different level of hierarchical Web cache. The specification of the test bed is shown in Table 2. Figure 2 represents that our content based hierarchical caching (CBHC) consumes less bandwidth for refreshing page contents in comparison to the current LCE, LCD and MCD caching strategies (as described in section 2). Figure 3 represents that the proposed framework consumes short time, due to categorization and
Table 2: Test-bed for Performance Evaluation

<table>
<thead>
<tr>
<th>PC</th>
<th>ROLE</th>
<th>CPU</th>
<th>MEMORY</th>
<th>OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Server</td>
<td>P-II</td>
<td>128 MB</td>
<td>WIN 2000</td>
</tr>
<tr>
<td>2</td>
<td>D-Cache</td>
<td>P-IV</td>
<td>1 GB</td>
<td>WIN 2000</td>
</tr>
<tr>
<td>3</td>
<td>P-Cache</td>
<td>P-II</td>
<td>128 MB</td>
<td>WIN 2000</td>
</tr>
<tr>
<td>4</td>
<td>S-Cache</td>
<td>P-IV</td>
<td>1 GB</td>
<td>WIN 2000</td>
</tr>
</tbody>
</table>

Figure 3: Time Consumption in Page Refreshment

Figure 4: Computation Overhead

Figure 2: Bandwidth Consumption in Page Refreshment
distribution of contents, in refreshing a Web page contents in comparison to refreshing the whole page at each node of the hierarchical Web cache. Figure 4 shows the computation overhead of our system. In the proposed architecture data has been distributed over different levels of hierarchy so, it will take some time in fetching the contents of a page from other levels. This has been shown in the graph.

8. CONCLUSION AND FUTURE WORK

In this paper, we have proposed a novel framework for the hierarchical Web caching. The framework categorizes the contents of a Web page and distribute accordingly to the various levels of hierarchy. The framework consumes less bandwidth and time in refreshing the page contents at various levels as shown in the above section. Moreover it avoids complete replication of a page contents and reduces the original server load. It balances the load of different levels of cache. In future we will extend the proposed framework to handle database-driven dynamic and multimedia contents.

References

152, USENIX Association, Jan. 1996.

Proc. of IEEE Conference on Computer Communications (INFOCOM ’99), 99. 126-
[19]. Mikhail Mikhailov, Craig E. Wills: Evaluating a new approach to strong web cache consistency with snapshots of collected content. WWW 2003: 599-608