REGULATION MECHANISM FOR CACHING IN PORTAL APPLICATIONS

Mehregan Mahdavi
Department of Computer Engineering, Guilan University, Rasht, Iran

John Shepherd, Boualem Benatallah
School of Computer Science and Engineering, Sydney, Australia

Keywords: Web Caching, Collaborative Caching, Portal, Regulation.

Abstract: Web portals are emerging Web-based applications that provide a single interface to access different data or service providers. Caching data from different providers at the portal can increase the performance of the system in terms of throughput and user-perceived delay. The portal and its providers can collaborate in order to determine the candidate caching objects. The providers allocate a caching score to each object sent to the portal. The decision for caching an object is made at the portal mainly based on these scores. However, the fact that it is up to providers to calculate such caching scores may lead to inconsistencies between them. The portal should detect these inconsistencies and regulate them in order to achieve a fair and effective caching strategy.

1 INTRODUCTION

The World Wide Web has already changed many aspects of life such as communication, education, business, shopping, and entertainment. It provides a convenient and inexpensive infrastructure for communicating and exchanging data between users and data sources. Users can search for information, products, and services, to use or buy. Web sites of universities, people’s home pages, yellow and white pages, on-line stores, flight reservation, hotel booking, and electronic banking are just some examples. These Web sites are referred to as providers. There are large number of such providers that provide the same sort of information, products, and services. Therefore, it can be time consuming for users to navigate through them in order to find what they need.

Web portals are emerging class of Web applications that provide a single interface for accessing different providers. The users only need to visit the portal instead of navigating through individual providers. In other words, they save time and efforts for users. Portals, such as Expedia (www.expedia.com) and Amazon (www.amazon.com) are examples of such applications.

Performance and in particular providing a fast response time is one of the critical issues that today’s Web applications must deal with. Previous research has shown that abandonment of Web sites dramatically increases with increase in response time (Zona Research Inc., 2001), resulting in loss of revenue by businesses. Nowadays, many Web sites employ dynamic Web pages by accessing a back-end database and formatting the result into HTML or XML pages. Accessing the database and assembling the final result on the fly is an expensive process and a significant factor in the overall performance of such systems. Server workload or failure and network traffic are other contributing factors for slow response times.

With the increasing use of the Web-enabled applications there is a need for better performance. Caching is one of the key techniques that addresses some of the performance issues of such applications. Caching can improve the response time. As a result, customer satisfaction is increased and better revenue for the portal and the providers is generated. In addition, network traffic and the workload on the providers’ servers are reduced. This in turn improves throughput and scalability and reduces hardware and software costs.

Caching a particular object at the portal depends on the available storage space, response time
than those who score higher. It may also result in less performance improvements are expected to be less scores get comparatively less cache space and their treatment of providers. As a result those who give lower cache space than others. This leads to unfair treatment. Although, all providers may use the same overall strategy to score their objects, the scores may not be consistent. In the absence of any regulation of cache-worthiness scores, objects from providers who give higher scores will get more chance to be cached, and such providers will get more cache space than others. This leads to unfair treatment of providers. As a result those who give lower scores get comparatively less cache space and their performance improvements are expected to be less than those who score higher. It may also result in less effective cache performance as a whole. To achieve an effective caching strategy, the portal should detect these inconsistencies and regulate the scores given by different providers.

The remainder of this paper is organized as follows: Section 2 provides an overview about Web caching. In Section 3 we explain the regulation mechanism used in a collaborative caching environment. Experimental results are presented in Section 4. Finally, some conclusions are presented in Section 5.

2 WEB CACHING BACKGROUND

Web caching has been studied extensively. Browser and proxy caches are the most common caching strategies for (static) Web pages. Caching dynamic Web pages has been studied in (Aberdeen Group, 2001; Chutney Technologies, 2001; Chutney Technologies, 2006; Akamai Technologies Corporate, 2001; Chidlovskii and Borghoff, 2000; Candan et al., 2001; Oracle Corporation, 2006; Oracle Corporation, 2006; Oracle Corporation, 2001a; Anton et al., 2002; Challenger et al., 1999; Dynamai, 2006; TimesTen Inc., 2002). Caching Web objects has already created a multi-million dollar business: Content Delivery/Distribution Network (CDN) (Oracle Corporation, 2006; Vakali and Pallis, 2003). Companies such as Akamai (Akamai Technologies Corporate, 2006) have been providing CDN services for several years. CDN services are designed to deploy edge servers at different geographical locations. Examples of edge servers include Akamai EdgeSuite (Akamai Technologies Corporate, 2006) and IBM WebSphere Edge Server (IBM Corporation, 2006).

Some applications may need a customized caching technique. Application-level caching is normally enabled by providing a cache API, allowing application writers to explicitly manage the cache to add, delete, and modify cached objects (Degenaro et al., 2001; Borvedt, 2004; Sun Microsystems, 2005; Apache Software Foundation, 2004). When considering caching techniques, a caching policy is required to determine which objects should be cached (Podlipnig and Boszormenyi, 2003; Balamash and Krunz, 2004; Cao and Irani, 1997; Datta et al., 2002; Aggrawal et al., 1999; Cheng and Kam-bayashi, 2000a; Young, 1991; Wong and Yeung, 2001). For rapidly changing data we might prefer not to cache them because of the space, communication or computation costs. Products such as Oracle Web Cache (http://www.oracle.com), IBM WebSphere Edge Server (http://www.ibm.com), and Dynamai (http://www.persistence.com) enable sys-
tem administrators to specify caching policies. Weave (Florescu et al., 2000; Yagoub et al., 2000) is a Web site management system which provides a language to specify a customized cache management strategy.

The performance of individual cache servers increases when they collaborate with each other by replaying each other’s misses. A protocol called Inter-cache Communication Protocol (ICP) was developed to enable querying other proxies in order to find requested web objects. (Li et al., 2001; Paul and Fei, 2000; Cheng and Kambayashi, 2000b; Fan et al., 2000; Rohm et al., 2001; Chandhok, 2000). In Summary Cache, each cache server keeps a summary table of the content of the cache at other servers. When a cache miss occurs, the server probes the table to find the missing object in other servers. It then sends a request only to those servers expected to contain the missing object (Fan et al., 2000). In Cache Array Routing Protocol (CARP) all proxy servers are included in an array membership list and the objects are distributed over the servers using a hash function (Microsoft Corporation, 1997).

3 REGULATION MECHANISM

In current systems, caching policies are defined and tuned by parameters which are set by a system administrator based on the previous history of available resources, access and update patterns. A more useful infrastructure should be able to provide more powerful means to define and deploy caching policies, preferably with minimal manual intervention. As the owners of objects, providers are deemed more eligible and capable of deciding objects for caching purpose.

A caching score (called cache-worthiness) can be associated to each object, determined by the provider of that object. The cache-worthiness score is sent by the provider to the portal in response to a request from the portal.

Cache-worthiness scores are determined by providers via off-line process which examines the provider’s server logs, calculates scores and then stores the scores in a local table. In calculating cache-worthiness, the providers consider parameters such as access frequency, update frequency, computation/construction cost and delivery cost. More details on the caching strategy are presented in (Mahdavi et al., 2003; Mahdavi et al., 2004; Mahdavi and Shepherd, 2004).

A typical cache-worthiness calculation would assign higher scores to objects with higher access frequency, lower update frequency, higher computation/construction cost, and higher delivery cost. However, each provider can have its own definition of these scores, based on its own policies and priorities. For example, a provider might choose not to process server logs for defining the scores. It might, for example, choose to let the system administrator assign some zero or non-zero values to objects.

Relying on the providers to calculate and assign caching scores may lead to inconsistencies between them. The following factors contribute to causing inconsistencies in caching scores among providers:

- Each provider uses a limited number of log entries to extract required information, and the available log entries may vary from one to another
- Providers may use other mechanisms to score the objects (they are “not required” to use the above approach)
- Malicious providers may claim that all of their own objects should be cached, in the hope of getting more cache space

To achieve a fair and effective caching strategy, the portal should detect these inconsistencies and regulate the scores given by different providers. For this purpose, the portal uses a regulating factor \(\lambda(m)\) for each provider and applies it to the cache-worthiness scores and uses the result in the calculation of the overall caching scores received from provider \(m\). This factor has a neutral value in the beginning and is adapted dynamically by monitoring the cache behavior. This is done by tracing false hits and real hits.

A false hit is a cache hit occurring at the portal when the object is already invalidated. False hits degrade the performance and increase the overheads both at portal and provider sites, without any outcome. These overheads include probing the cache validation table, generating validation request messages, wasting cache space, and probing cache look-up table.

A real hit is a cache hit occurring at the portal when the object is still fresh and can be served by the cache. The performance of the cache can only be judged by real hits.

The portal monitors the performance of the cache in terms of tracing real and false hits and dynamically adapts \(\lambda(m)\) for each provider. For those providers with higher ratio of real hits, the portal upgrades \(\lambda(m)\) by a small amount \(\delta_1\). The new value for \(\lambda(m)\) is calculated by adding a small value to it. Therefore, all the cache-worthiness scores from that provider are treated as being higher than before. Choosing a small increment (close to 0) ensures that the increase is done gradually. Recall that we impose an upper bound of 1 on cache-worthiness scores. The new value of \(\lambda(m)\) will be:
\[
\lambda(m) \leftarrow \lambda(m) + \delta_1
\]  

(1)

For those providers with higher ratio of false hits, the portal downgrades \(\lambda(m)\) by a small amount \(\delta_2\). The new value for \(\lambda(m)\) is calculated by decreasing a small value from it. Therefore, all the cache-worthiness scores from that provider are treated as lower scores. Choosing a small decrement (close to 0) ensures that the decrease is done gradually. Recall that we impose a lower bound of 0 on cache-worthiness scores. The new value of \(\lambda(m)\) will be:

\[
\lambda(m) \leftarrow \lambda(m) - \delta_2
\]  

(2)

A high false hit ratio for a provider \(m\), indicates that the cache performance for that particular provider is not utilized. That is because the cached objects for that provider are not as worthy as they should be. In other words, the provider has given higher cache-worthiness scores to its objects. This can be resolved by downgrading the scores from that provider and treat them as they were lower.

Unlike a high false hit ratio for a provider \(m\), indicates that the cache performance for this provider is good. Therefore, provider \(m\) is taking good advantage of the cache space. Upgrading the cache-worthiness scores of provider \(m\) results in more cache space being assigned to this provider. This ensures fairness in the cache usage based on how the cache is utilized by providers. The fair distribution of cache space among providers will also result in better cache performance. The experimental results confirm this claim.

4 EXPERIMENTS

In order to evaluate the performance of the collaborative caching strategy, a test-bed has been built. This test-bed enables us to simulate the behavior of a business portal with different number of providers, message sizes, response time, update rate, cache size, etc. We examine the behavior of the system under a range of different scenarios.

The performance results show that the collaborative caching strategy (i.e., CacheCh) outperforms other examined caching strategies by at least 22% for throughput, 24% for network bandwidth usage and 18% for average access time. Examined caching strategies include Least Recently Used (LRU), First In First Out (FIFO), Least Frequency Used (LFU), Size (SIZE), Size Adjusted LRU (LRU-S), and Size Adjusted and Popularity Aware LRU (LRU-SP).

To address the issue of inconsistencies, a regulation factor is assigned to each provider. Every provider’s cache-worthiness score is multiplied by the corresponding factor. Therefore, providers whose scores are low should have a high regulation factor and vice versa. The regulation factor changes over time by monitoring false and real hit ratio.

For this purpose, the providers in the simulation are set up in such a way that first one deliberately overestimates, second one underestimates, and third one produces the standard cache-worthiness score. The same pattern applies for subsequent providers. In other words, one third of providers overestimate, one third underestimate, and for the remaining one third the normal cache-worthiness score is considered. Each provider was initially given a regulation factor of 1, so that each cache-worthiness score from them was not modified.

False and real hit ratio were used to downgrade or upgrade the regulation factor. However, using real hit ratio over the occupied cache space by each provider for upgrading the regulation factor was the most successful among all the variations used. Using real hit ratio by itself did not produce the desired results, as the performance of the cache for a provider depends both on the real hit ratio and the cache space occupied by the provider.

Providers were monitored to see if the regulation factor was moving in such a way as to separate the three groups of providers so that the underestimating providers consistently had the highest factor, followed by the accurately estimating provider, with the overestimating provider having the lowest regulation factor. Figure 1 shows the changes in regulation factor for different groups of providers. One provider from each group is used in the Figure. However, all providers in each group show similar results. The results demonstrate that the regulation factor does separate the providers accordingly.

![Figure 1: Regulation Factor.](image)

When upgrading or downgrading the regulation factor, we use small values \(\delta_1\) and \(\delta_2\) by which \(\lambda(m)\) is upgraded or downgraded. By choosing very small
value for $\delta_1$ and $\delta_2$, it takes a long time for the system to adjust itself. On the other hand, choosing large value for $\delta_1$ and $\delta_2$ makes the regulation factor fluctuate unnecessarily. Choosing an appropriate value makes the system adjust itself in a fair amount of time. For this purpose we have examined different values for $\delta_1$ and $\delta_2$.

\[
\Delta_i(CW) = CW_{i+1} - CW_i \quad (3)
\]

\[
\delta_1 = f_1 \times \Delta(CW) : 0 < f_1 \leq 1 \quad (4)
\]

\[
\delta_2 = f_2 \times \Delta(CW) : 0 < f_2 \leq 1 \quad (5)
\]

Where:

- $CW_{i,j}$: Cache-worthiness score of object $O_j$ at provider $P_i$
- $CW_i$: Average value of cache-worthiness scores at provider $P_i$

Smaller values for $f_1$ and $f_2$ make the adjustment smoother, but more timely. The experiment result show that any value for $f1$ and $f2$ in the range of $0 < f_1, f_2 \leq 1$ will generate the expected results. The results in this experiment are generated based on $f_1 = 0.1$ and $f_2 = 0.1$. When $\Delta(CW) = 0$, although very unlikely, the regulation factor will be zero. In other words, in this special case the regulation process will stay unchanged. However, in the next interval, when $\Delta(CW)$ is calculated again the regulation process will resume. The value for $\Delta(CW)$ is calculated using available objects in the cache. Our experiments show that even using a subset of cached objects to generate $\Delta(CW)$ will give the same results and an estimation of the value, in case the overhead is an issue, will generate the desired results.

According to the experiments regulation results in improvement in the throughput. The results are shown in Table 1. The resulted throughput after regulation is 305 compared to 278 which is 10% improvement in throughput.

Table 1: Throughput.

<table>
<thead>
<tr>
<th></th>
<th>Throughput</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoReg</td>
<td>278</td>
</tr>
<tr>
<td>Reg</td>
<td>305</td>
</tr>
</tbody>
</table>

Average access time for individual providers is improved as a result of better utilization of cache space, as shown in Table 2. However, those providers that take better advantage of cache, show better improvement (i.e., those with higher real hit ratio). In our example, these are providers that underestimate and are shown as UnderEst, providers that overestimate (i.e., OverEst) result in less improvement. In other words, the improvement in average access time for individual providers is in proportion with their utilization of cache space. Total average access time is also improved as a result of regulation process and better utilization of cache space (i.e., 6.59 compared to 7.26) as shown in Table 3.

Table 2: Average access times for individual providers.

<table>
<thead>
<tr>
<th></th>
<th>UnderEst</th>
<th>NormalEst</th>
<th>OverEst</th>
</tr>
</thead>
<tbody>
<tr>
<td>NoReg</td>
<td>7.54</td>
<td>7.21</td>
<td>7.04</td>
</tr>
<tr>
<td>Reg</td>
<td>6.63</td>
<td>6.58</td>
<td>6.55</td>
</tr>
</tbody>
</table>

5 CONCLUSION

In this work, we discussed how a collaborative caching strategy can overcome the limitations of current systems in providing an effective caching strategy in portal applications. We addressed the issue of heterogeneous caching policies by different providers and introduced a mechanism to deal with that. This is done by tracing the performance of the cache and regulating the scores from different providers. As a result, better performance for individual providers is achieved and the performance of the cache as a whole is improved.

REFERENCES


