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

Mahdavi M., Shepherd J. and Benatallah B. (2007).
REGULATION MECHANISM FOR CACHING IN PORTAL APPLICATIONS.
In *Proceedings of the Second International Conference on Software and Data Technologies - PL/DPS/KE/WsMUSE*, pages 152-157
DOI: 10.5220/0001340701520157

(QoS) requirements, access and update frequency of objects (Kossmann and Franklin, 2000). Available caching systems enable system administrators to specify caching policies. This is done mainly by including or excluding objects or object groups (e.g., objects with a common prefix in the URI) to be cached, determining expiry date for caching objects or object groups, and etc. Server logs (i.e., access log, and database update log) are also used to identify objects to be cached (Oracle Corporation, 2001b; IBM Corporation, 2006; Dynamai, 2006; Florescu et al., 2000; Yagoub et al., 2000).

Existing caching approaches have examined caching in a general setting and can provide some benefit to portals. However, portals have distinctive properties to which existing techniques cannot be easily adapted and used. Most importantly, the portal and providers are managed by different organizations and administrators. Therefore, the administrator of portal does not normally have enough information to determine caching policies for individual providers. Moreover, since the portal may be dealing with a (large) number of providers, determining the best objects for caching manually or by processing logs is impractical. On one hand, an administrator cannot identify candidate objects in a dynamic environment where providers may join and leave the portal frequently. On the other hand, keeping and processing access logs in the portal is impractical due to high storage space and processing time requirements. Also, database update logs are not normally accessible by the portal.

The portal and its providers can collaborate in order to provide a more effective caching strategy. A caching score (called **cache-worthiness**) is associated to each object, determined by the provider of that object. It represents the usefulness of caching this object at the portal. Larger values represent more useful objects for caching. The cache-worthiness score is sent by the provider to the portal in response to a request from the portal (Mahdavi et al., 2003; Mahdavi et al., 2004; Mahdavi and Shepherd, 2004).

The fact that it is up to providers to calculate cache-worthiness scores may lead to inconsistencies between them. 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, 2006; Chidlovskii and Borghoff, 2000; Candan et al., 2001; 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; Bortvedt, 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 Kambayashi, 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 system 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 replying each other's misses. A protocol called *Intercache 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 an 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 δ_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:

$$\lambda(m) \leftarrow \lambda(m) + \delta_1 \quad (1)$$

For those providers with higher ratio of false hits, the portal downgrades $\lambda(m)$ by a small amount δ_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:

$$\lambda(m) \leftarrow \lambda(m) - \delta_2 \quad (2)$$

A high false hit ratio for a provider *m*, indicates that the cache space 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.

Unlikely, a high real 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., `CacheCW` 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.

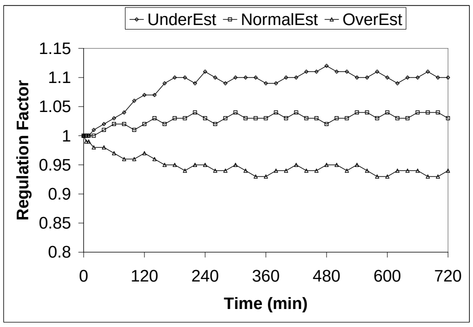

Figure 1: Regulation Factor.

When upgrading or downgrading the regulation factor, we use small values δ_1 and δ_2 by which $\lambda(m)$ is upgraded or downgraded. By choosing very small value for δ_1 and δ_2, it takes a long time for the system to adjust itself. On the other hand, choosing large value for δ_1 and δ_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 δ_1 and δ_2.

$$\Delta_i(\overline{CW}) = \overline{CW_{i+1}} - \overline{CW_i} \tag{3}$$

$$\delta_1 = f_1 \times \overline{\Delta(\overline{CW})} : 0 < f_1 \leq 1 \tag{4}$$

$$\delta_2 = f_2 \times \overline{\Delta(\overline{CW})} : 0 < f_2 \leq 1 \tag{5}$$

Where:

$CW_{i,j}$:	Cache-worthiness score of object O_j at provider P_i
$\overline{CW_i}$:	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 f_2 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 $\overline{\Delta(\overline{CW})} = 0$, 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 $\overline{\Delta(\overline{CW})}$ is calculated again the regulation process will resume. The value for $\overline{\Delta(\overline{CW})}$ is calculated using available objects in the cache. Our experiments show that even using a subset of cached objects to generate $\overline{\Delta(\overline{CW})}$ 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.

	Throughput
`NoReg`	278
`Reg`	305

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.

	`UnderEst`	`NormalEst`	`OverEst`
`NoReg`	7.54	7.21	7.04
`Reg`	6.63	6.58	6.55

Table 3: Total average access time.

	Average Access Time
`NoReg`	7.26
`Reg`	6.59

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

Aberdeen Group (2001). Cutting the Costs of Personalization With Dynamic Content Caching. http://www.chutneytech.com/tech/aberdeen.cfm. An Executive White Paper.

Aggrawal, C., Wolf, J. L., and Yu, P. S. (1999). Caching on the World Wide Web. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 11(1):94–107.

Akamai Technologies Corporate (2006). http://www.akamai.com.

Anton, J., Jacobs, L., Liu, X., Parker, J., Zeng, Z., and Zhong, T. (2002). Web Caching for Database Aplications with Oracle Web cache. In *ACM SIGMOD Conference*, pages 594–599. Oracle Corporation.

Apache Software Foundation (2004). JCS and JCACHE (JSR-107). http://jakarta.apache.org/turbine/jcs/index.html.

Balamash, A. and Krunz, M. (2004). An Overview of Web Caching Replacement Algorithms. In *IEEE Comunications Surveys and Tutorials*, volume 6, pages 44–56.

Bortvedt, J. (2004). Functional Specification for Object Caching Service for Java (OCS4J), 2.0. http://jcp.org/aboutJava/communityprocess/jsr/cacheFS.pdf.

Candan, K. S., Li, W.-S., Luoand, Q., Hsiung, W.-P., and Agrawal, D. (2001). Enabling Dynamic Content Caching for Database-Driven Web Sites. In *ACM SIGMOD Conference*, pages 532–543.

Cao, P. and Irani, S. (1997). Cost-Aware WWW Proxy Caching Algorithms. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, pages 193–206.

Challenger, J., Iyengar, A., and Dantzig, P. (1999). A Scalable System for Consistently Caching Dynamic Web Data. In *IEEE INFOCOM*, pages 294–303.

Chandhok, N. (2000). Web Distribution Systems: Caching and Replication. http://www.cis.ohio-state.edu/ jain/cis788-99/web-caching/index.html.

Cheng, K. and Kambayashi, Y. (2000a). LRU-SP: A Size-Adjusted and Popularity-Aware LRU Replacement Algorithm for Web Caching. In *IEEE Compsac*, pages 48–53.

Cheng, K. and Kambayashi, Y. (2000b). Multicache-based Content Management for Web Caching. In *WISE*.

Chidlovskii, B. and Borghoff, U. (2000). Semantic caching of Web queries. *VLDB Journal*, 9(1):2–17.

Chutney Technologies (2001). Dynamic Content Acceleration: A Caching Solution to Enable Scalable Dynamic Web Page Generation. In *SIGMOD Conference*.

Chutney Technologies (2006). http://www.chutneytech.com.

Datta, A., Dutta, K., Thomas, H. M., VanderMeer, D. E., and Ramamritham, K. (2002). Accelerating Dynamic Web Content Generation. *IEEE Internet Computing*, 6(5):27–36.

Degenaro, L., Iyengar, A., and Ruvellou, I. (2001). Improving Performance with Application-Level Caching. In *International Conference on Advances in Infrastructure for Electronic Business, Science, and Education on the Internet (SSGRR)*.

Dynamai (2006). http://www.persistence.com.

Fan, L., Cao, P., and Broder, A. (2000). Summary Cache: A Scalable Wide-Area Web Cache Sharing Protocol. In *IEEE/ACM Transactions on Networking*, volume 8.

Florescu, D., Yagoub, K., Valduriez, P., and Issarny, V. (2000). WEAVE: A Data-Intensive Web Site Management System. In *The Conference on Extending Database Technology (EDBT)*.

IBM Corporation (2006). http://www.ibm.com.

Kossmann, D. and Franklin, M. J. (2000). Cache Investment: Integrating Query Optimization and Distributed Data Placement. In *ACM TODS*.

Li, D., Cao, P., and Dahlin, M. (2001). WCIP: Web Cache invalidation Protocol. http://www.ietf.org/internet-drafts/draft-danli-wrec-wcip-01.txt.

Mahdavi, M., Benatallah, B., and Rabhi, F. (2003). Caching Dynamic Data for E-Business Applications. In *International Conference on Intelligent Information Systems (IIS'03): New Trends in Intelligent Information Processing and Web Mining (IIPWM)*, pages 459–466.

Mahdavi, M. and Shepherd, J. (2004). Enabling Dynamic Content Caching in Web Portals. In *14th International Workshop on Research Issues on Data Engineering (RIDE'04)*, pages 129–136.

Mahdavi, M., Shepherd, J., and Benatallah, B. (2004). A Collaborative Approach for Caching Dynamic Data in Portal Applications. In *The Fifteenth Australasian Database Conference (ADC'04)*, pages 181–188.

Microsoft Corporation (1997). Cache Array Routing Protocol and Microsoft Proxy Server 2.0. http://www.mcoecn.org/WhitePapers/Mscarp.pdf. White Paper.

Oracle Corporation (2001a). Oracle9i Application Server: Database Cache. Technical report, Oracle Corporation, http://www.oracle.com.

Oracle Corporation (2001b). Oracle9iAS Web Cache. Technical report, Oracle Corporation, http://www.oracle.com.

Oracle Corporation (2006). http://www.oracle.com.

Paul, S. and Fei, Z. (2000). Distributed Caching with Centralized Control. In *5th International Web Caching and Content Delivery Workshop*.

Podlipnig, S. and Boszormenyi, L. (2003). A Survey of Web Cache Replacement Strategies. In *ACM Computing Surveys*, volume 35, pages 374–398.

Rohm, U., Bohm, K., and Schek, H.-J. (2001). Cache-Aware Query Routing in a Cluster of Databases. In *ICDE*.

Sun Microsystems (2005). JSRs: Java Specification Requests. http://www.jcp.org/en/jsr/overview.

TimesTen Inc. (2002). Mid-Tier Caching. Technical report, TimesTen Inc., http://www.timesten.com.

Vakali, A. and Pallis, G. (2003). Content Delivery Networks: Status and Trends. *IEEE Internet Computing*, pages 68–74.

Wong, K. Y. and Yeung, K. H. (2001). Site-Based Approach to Web Cache Design. *IEEE Internet Computing*, 5(5):28–34.

Yagoub, K., Florescu, D., Valduriez, P., and Issarny, V. (2000). Caching Strategies for Data-Intensive Web Sites. In *Proceedings of 26th International Conference on Very Large Data Bases (VLDB)*, pages 188–199, Cairo, Egypt.

Young, N. E. (1991). The k-Server Dual and Loose Competitiveness for Paging. *Algorithmica*, 11(6):525–541.

Zona Research Inc. (2001). Zona Research Releases Need for Speed II. http://www.zonaresearch.com/info/press/01-may03.htm.