RC22826 (W0306-106) June 20, 2003
Computer Science

IBM Research Report

Insights on the Performance of Cache Management Policies for Fragment-Assembling Caches

Daniela Rosu, Xindian Long
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 704
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

Insights on the Performance of Cache Management Policies for Fragment-Assembling Caches

Daniela Rosu
IBM T.J. Watson Research Center
P.O. Box 704
Yorktown Heights NY 10598, USA
drosu@us.ibm.com

Xindian Long*
CMU
xlong@andrew.cmu.edu

Abstract

The Edge Side Includes (ESI) standard enables Web proxy caches to deliver dynamic web content, such as personalized portal pages and e-business queries, by assembling it on demand from page templates and content fragments retrieved from local caches or remote servers.

This paper investigates whether ESI caches can benefit from exploiting the structure of the composed pages. Toward this end, we view the space of dynamic pages as a partition defined by page templates; the fragments used for each template are grouped in classes, based on the ESI tags that define them. Thus a dynamic page is composed from a page template and one fragment from each of the associated classes.

We consider two components of the cache management infrastructure: the replacement policy and the space allocation. Our experiments show that replacement policies that account for page-structure dependencies can improve hit ratios in comparison to Greedy-Dual Size-Frequency policy. Similarly, partitioning the cache space among fragment classes can improve hit ratios in comparison to sharing the entire cache space among all fragment classes, as traditionally used. The study is conducted through simulation with simple synthetic workloads.

*Work done during a Summer Internship with IBM T.J. Watson Research Center.

1 Introduction

Driven by the need to provide personalized content and time sensitive information, Web content providers are producing an increasing volume of dynamic content. This trend raises significant challenges for the supporting Web infrastructures due to higher demand for computation and network resources, and to increased interactions between application and database servers.

Addressing these problems, previous research proposed to build dynamic content by assembling content components, called fragments, which can be cached and selectively updated [6, 19]. Numerous studies show that caching fragments at application servers reduces the resource consumption on origin Web site infrastructures [11, 22].

More recent proposals advocate the offload of dynamic content composition onto Web proxies [5, 9, 10, 18, 23] or Web browsers [20]. Provided with an appropriate specification of the dynamic page structure, such as an ESI template [7], a Web proxy or browser can identify the component fragments, retrieve them from the local cache or origin sites, and, finally, assemble and deliver the content to the client.

Related to this approach, previous research has addressed the problems of dynamic content representation [7, 10] and of consistency management for the cached fragments [23, 18]. In this study, we address the problem of cache management, considering the issues of replacement policy and space allocation. The problem is relevant because, for acceptable performance, Web proxy caches that as-

semble fragment-based content need to maximize hit ratios and to rely heavily on main-memory caches [8, 9]. Therefore, the available cache sizes are typically small, which makes it hard to achieve acceptable hit ratios, in particular for sites with large content spaces, such as the sport-event sites.

Our study is based on the *Fragment Class* (FC) model of dynamic content, designed to capture the correlations among content fragments that derive from the structure of dynamic pages. This model extends the previously proposals [21, 23], which consider each dynamic page as an independent entity, obtained through the composition of a container and several objects. In contrast, in the FC model, the universe of dynamic content is composed of *groups* of pages with similar fragmentation features. All pages in a group are derived from the same HTML template. Further, each fragment tag in the template is associated with a *fragment class*, which is a set of fragments generated by the same method but with different parameters. To compose a dynamic object one has to retrieve the template and one fragment from each of the classes associated with the template. The FC model can describe the characteristics of a wide range of dynamic content by appropriately setting the characteristics of templates and fragment classes, like number of objects and object popularity model.

With respect to cache replacement, this study proposes a new policy, called 'Greedy-Dual Size-Class Relative Frequency' (GDS-CRF), which exploits the correlations that derive from the structure of fragment-based content. This policy can provide better hit ratios than Greedy-Dual Size-Frequency (GDS-F) [1], a cache replacement policies considered optimal for static content. The limitation of GDS-F and similar policies [12] derives from the use of access frequency as a measure for the relative importance of objects. This approach is appropriate for static-content caches, in which each client request translates into a single cache request. However, for fragment-based content, access frequency is not a good measure for the relative importance of fragments. For instance, a fragment X included from two templates tends to have a higher access frequency than fragments Y and Z included from each of the two templates. However, fragment X does not bring more value to the cache than either of fragments Y and Z.

With respect to cache space allocation, this study demonstrates that by partitioning the cache space among fragment classes one can achieve better performance than by the traditional sharing the entire cache space among all classes.

In this simulation-based study, we use several dynamic content pages modeled with the FC model. Our results represent only the initial steps towards understanding the cache performance in fragment-assembling proxies. A limitation of our study, which we expect to address in the future, is the lack of realistic content models, and this derives from the scarce information available in the literature on workloads for fragment-assembling CDNs or reverse proxies.

The remainder of this study is organized as follows. Section 2 presents the related work. Section 3 presents the details of the FC model. Section 4 describes the simulator for FC workloads used in this study. Section 5 presents the experimental results. Finally, Section 6 summarizes our contributions and highlights several future-work items.

2 Related Work

The problem of improving the performance of dynamic content Web sites has received significant attention during the past several years. The most advocated solutions are based on caching dynamic content in various representations, including the whole pre-computed content [11, 24], fragments ready to be assembled [18], or intermediary database query results[16]. This study is related to work in the area of fragment-based decomposition and caching.

Fragment-based Decomposition. Drawing from results in the area of Web site content management [6, 13, 24], previous work advocates the idea of enabling Web proxies to compose dynamic content on demand, by assembling a collection of component objects (fragments) [5, 9, 10, 17, 23]. One approach is to provide the Web proxy with an explicit description of the dynamic pages that it can compose and serve to clients. Tag-based specifications, including ESI [7], HPP [10], and others [17], allow to describe a dynamic page as composed of a page

template and a set of content fragments; the fragment specification may be determined by request parameters. Upon serving a request, a Web proxy retrieves the page template from the cache or origin server, it processes the included tags [7, 17] or macros [10] in the context of current request parameters, and it issues requests to the local cache or remote servers to retrieve the identified fragments.

Another approach is the Active Cache paradigm [5], in which dynamic content is specified implicitly, by a URL-specific applet. A cache applet can retrieve request parameters, access local databases, and if necessary, interact with the origin server. The applet output represents the requested dynamic content.

In this study, we do not restrict the specification paradigm. However, we assume that some parameters of the content fragmentation model, such as the mapping of fragments to classes, can be provided to the caching infrastructure.

Modeling Dynamic Content Characteristics. Several studies [21, 23] attempt to model the characteristics of fragment-based content with respect to object size and update rate distributions. These studies use the home pages of several popular sites and identify the segments of content that could represent fragments by tracing the changes across a sequence of page versions. While providing insights on the characteristics of dynamic content across several sites, both studies omit to explore the characteristics of dynamic content within a single site.

We submit that one has to consider the structure of the entire dynamic content of a Web site in order to appropriately evaluate the implications of proxy-level fragment assembling. This is because we believe fragment assembling is more likely to be deployed at reverse proxies and CDNs, which handle complete sites, rather than at forward proxies, which handle individual pages.

The content space of an entire site is considered in [22]. With focus on motivating the need for fragment cacheability, the study takes a global view to the fragment space, focusing on statistical characteristics of client requests and fragment size distribution across all channels. However, the study does not model channel characteristics, like popularity and size distributions.

The *Fragment Class* model proposed in this paper allows us to describe the complete content space of a site in a way that is consistent with the page and fragment generation process. In addition, the model allows us to represent content from one or more sites, as necessary for reverse proxy and CDN studies. The FC model can capture a wide variety of dynamic content, including portal content [22], or content with frequently updated fragments [21, 23].

Proxy Cache Management. A significant body of research has addressed the problem of cache replacement in Web proxy caches, analyzing the effect of various caching policies on object and byte hit ratios. The better performing methods extend the LRU object utility value based on access recency to include object size [4] and access frequency components [1, 12]. This approach improves object hit ratios. Byte-hit ratios and system-specific cost metrics, such as download latency, are addressed by adding corresponding cost functions to the utility value.

All of the previous studies of cache replacement policies address the problem in a universe of static objects. Applied to fragment-based content, these policies consider the page templates and fragments are independently accessed entities. Our study demonstrates that replacement policies that account for the page structure and fragment-class characteristics can result in better hit ratios than traditional policies.

Previous studies have also proposed cache replacement policies that differentiate among the cached objects. For instance, [14] proposes a policy in which origin servers are assigned different weights for caching, and these weights reflect in the utility value of their objects: the higher the server weight, the larger the bias towards maintaining its objects in the cache. Similarly, [15] proposes a control-theoretical method for providing differentiated service to the set of origin servers represented in the cache; the solution is based on partitioning the cache space among origin servers, with partition sizes adjusted dynamically according to the observed hit rates.

Different from these studies, we consider cache replacement and space allocation policies that differentiate objects based on the structural characteristics of the dynamic content rather than on the con-

tent affiliation to particular servers.

3 Fragment Class Model

The *Fragment Class* (FC) model captures the characteristics of dynamic content in an approach consistent with the page and fragment generation process.

More specifically, in the FC model, a collection of dynamic pages is defined by three components: (1) a set of page templates, (2) a collection of sets of content fragments, called *fragment classes*, and (3) a mapping between page templates and fragment classes. For instance, for an ESI-based page template, each ESI tag corresponds to a fragment class, and the mapping associates these classes with the template.

Each dynamic page is assembled from a template and one fragment from each of the classes associated with the template through the mapping.

A fragment class can be associated with several templates. For instance, distinct personalized portal templates, like My.Yahoo pages, can refer to a fragment class of common interest for their owners, like the Technology News.

In order to characterize a dynamic workload, the templates and the fragment classes have several attributes. Template attributes include (1) size and (2) request probability. Fragment class attributes include (1) number of fragments, (2) distribution of fragment sizes, (3) distribution of fragment popularity, and (4) distribution of computation overheads. For studies that address the cache consistency problem, templates and fragment classes can be associated with an update model.

We assume that the likelihood of selecting a fragment from the class is identical for all the including templates; i.e., the distribution of fragment popularity in a class depends on the class and not on the templates associated with the class. Also, we assume that fragment classes do not share objects.

Dynamic content with a hierarchy of embedded fragments is represented by flattening the hierarchy to one page template and considering that all fragment classes are associated with the template, irrespective the levels of the corresponding ESI tags.

The FC parameters allow us to characterize a wide range of dynamic content spaces. For instance, for news sites like CNN, one can have a template for the main page, one template for each per-section main pages, e.g., health main page, and one or more templates for article pages. The template of the main page has several fragments, but the corresponding classes have only one object. Article-page templates have several fragments as well, but the class sizes are much larger. For portal sites, like My.Yahoo and the Web site analyzed in [22], one can have several templates, one for each combination of content sources. For instance, for My.Yahoo, each content source corresponds to a fragment class; some classes can have only one item, like the 'Health Features' class, some can have several items, like the 'Horoscope' class, while others can have numerous items, like the 'Weather' class.

4 Experimental Methodology

This study is conducted through simulations with synthetic workloads. The content space is defined by the FC model, described in Section 3. We rely on synthetic workloads because of the lack of realistic traces or models that provide the information necessary for the FC model. Although [22] reports the number and size range of objects in each class fragment, it gives no information about the popularity distributions.

Our simulator can use an FC model specified explicitly or one generated from a set of input parameters including: number of templates, template size distribution, distribution of fragment classes per template, distribution of number of shared classes among templates. For the experiments reported in this paper, we use an explicit specification.

The size and popularity distributions for a fragment class can be defined by several types of distributions. For these experiments, all fragments in a class have the same size, and the popularity distribution is Zipf.

The simulator works with traces computed offline or online. The use of offline traces is motivated by our intent to integrate temporal locality for fragment requests. We have experimented with the methods integrated in ProWGen [3] and SURGE

[2], but could not achieve acceptable distribution of fragment IDs across the request series. As a consequence, for this study we ignore temporal locality and use online generated traces.

For each experiment, the simulator reports several performance metrics including (object) hit ratio, complete-page hit ratio, per-class hit ratios, and distribution of fragment misses per request. Complete-page hit (CPH) ratio is defined as the ratio of pages requests for which all fragments are available in the local cache. The measurements start after the amount 'serviced' content reaches a specified threshold. In our experiments, this threshold is equal to the total content space.

Given our interest in performance of cache replacement policies, the simulator does not model client request rates and content updates. We assume that cached content is always valid.

5 Experimental Results

In this study, we focus on two components of the cache management, the replacement policy and the space allocation policy. The replacement policy is used by the cache to decide which objects can be discarded when space is needed to accommodate new objects. In Section 5.1, we propose and evaluate a new replacement policy, called *Greedy-Dual Size Class-Relative Frequency (GDS-CRF)*. This policy exploits information about the structure of dynamic content in order to alleviate the limitations with respect to fragment-based content specific to policies like GDS-F [1].

The space allocation policy determines how the cache space is divided among various categories of cacheable objects. Most studies related to cache management do not consider the space allocation policy, assuming that the cache manager uses the entire cache space to store all of the objects, independent of their content provider or access characteristics. In Section 5.2, we evaluate the benefits of partitioning the cache among fragment classes.

Dynamic Content Configurations. In this study, we use three dynamic content configurations. Table 1 presents their FC models.

All configurations have two templates and three fragment classes. Each template is accessed with 50% probability and includes two fragment classes. Both templates include the class C0 of the corresponding configuration.

Configurations differ by the number and size of objects the fragment class; object popularity within each class is defined by a Zipf distribution with coefficient 0.3. In configuration *FreqLarge (frequent-is-large)*, the objects in the most frequently accessed class, $C0$, are significantly larger than the objects in the other two classes. In configuration *FreqMany (frequent-is-many)*, the most frequently accessed class, C0, has significantly more objects than other classes, which results much smaller access probabilities for its objects. Finally, in configuration *FreqSmall (frequent-is-small)*, the objects in the most frequently accessed class, C0, are significantly smaller than the objects in other classes.

5.1 Implications of Replacement Policy

In traditional static-content caches, in which one client request translates to a single cache object, the best replacement policies use an object value function defined by a combination of access probability, size, and retrieval cost. For instance, the GDS-F [1] value of an object o is defined by: $GDS-F(o) = \frac{acces(o) \cdot cost(o)}{size(o)}$.

This type of value function models the intuition that the system gets more benefit by caching the objects with the largest access frequency per size unit. This matches the need of static content caches because it maximizes the number of client requests that can be served with cached content.

For fragment-assembling caches, cache accesses are no longer independent, as one client request translates to several cache objects. We identify two limitations of the value model used for static-content caches when applied to fragment-based content.

The first limitation is related to *class access from multiple templates*. Let us consider a content configuration in which one fragment class is included by several templates, while other fragment classes included by these templates are included by only one template. The configurations considered in this study match this profile, because class C0 is included by two templates while classes C1 and C2 are included by only one template. The intuition

Table 1: Dynamic Content Configurations.

Configuration	Fragment Classes				Templates			
	Id	Number Objs.	Size KB	Zipf Coeff.	Id	Access Prob.	Size KB	Classes
FreqLarge	C0	2000	10	0.3	B0	0.5	2	C0, C1
'frequent-is-large'	C1	500	1	0.3	B1	0.5	1	C0, C2
	C2	2000	2	0.3				
FreqMany	C0	10000	1	0.3	B0	0.5	2	C0, C1
'frequent-is-many'	C1	500	1	0.3	B1	0.5	1	C0, C2
	C2	2000	10	0.3				
FreqSmall	C0	2000	1	0.3	B0	0.5	2	C0, C1
'frequent-is-small'	C1	500	9	0.3	B1	0.5	1	C0, C2
	C2	2000	10	0.3				

is that objects in class C0 yield the same benefit to the cache as objects in classes C1 and C2, when the differences in size and intra-class popularity are ignored. However, in the static-content model, objects in class C0 are considered more valuable than objects in classes C1 and C0.

The second limitation of the GDS-F value model is related to *intra-class access probabilities*. Let us consider a template that includes two classes with very different number of objects, like template B0 in configuration FreqSmall. Objects in the less populated class tend to have higher access frequencies than objects in more populate class, like class C2 vs. class C0 in configuration. By the static-content model, C0 yields much lower value to the cache than C2, as even its most popular objects have lower values than almost all objects in C2. However, the intuition is that both classes yield a comparable benefit to the cache.

To address these limitations, we consider a value function which discounts an object's access frequency relative to the access frequency in the class. We call this value function *Greedy-Dual Size Class-Relative Frequency (GDS-CRF)*. Given an object o, the function is defined by $GDS - CRF(o) = \frac{access(o)}{max\_access\_class(o)} \frac{cost(o)}{size(o)}$, where $max\_access\_class(o)$ is the maximum of $access$ for all objects in the same class as o.

This function addresses the limitation related to access from multiple templates by eliminating the related bias in access frequency. Also, it addresses the limitation related to intra-class accesses by requiring that objects be accessed significantly more frequently relative to their class in order to achieve a much larger value than objects in other classes.

Figures 1- 3 present the (object) hit ratios (HIT) and the complete-page hit ratios (CPH) when varying the available caches space for all the configurations, when using GDSF, GDS-CRF, and LRU. The available cache space is expressed as a fraction of the space required to accommodate all of the objects in a configuration. Figures 4-6 presents the per-class hit ratios for GDSF and GDS-CRF.

The plots illustrate that both GDSF and GDS-CRF provide better performance than LRU for all configurations. Further, GDS-CRF benefits FreqMany and FreqLarge, but not FreqSmall. For FreqMany, Figure 5 illustrates that the most numerous class, C0, observes a better hit ratios because of discounted value of objects in class C2. For FreqLarge (see Figure 4), class C2 observes larger hit ratio with GDS-CRF due to a decrease in value for objects in class C0. For FreqSmall, the lower hit ratio of GDS-CRF is mostly due to the fact that objects in class C1 observe a discount relative to the objects in class C2 (see Figure 6).

For all configurations, the policy with the best hit ratio provides also the best complete-page hit ratio.

Overall, these experiments show that accounting for fragment-class characteristics helps improve the overall hit ratio in comparison to GDSF for some types of dynamic content configurations. Unfortu-

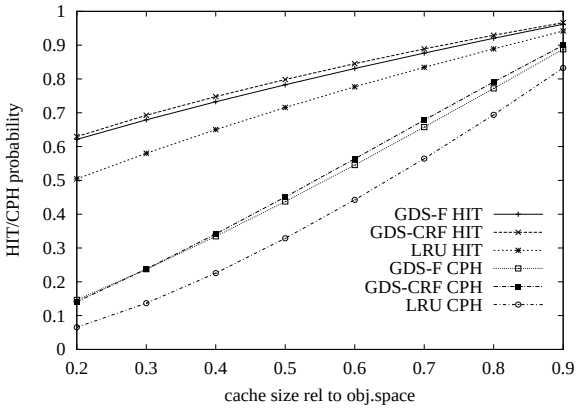

Figure 1: FreqLarge: HIT and CPH ratios.

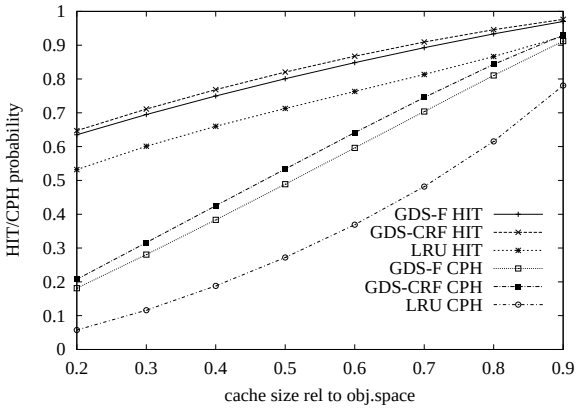

Figure 2: FreqMany: HIT and CPH ratios.

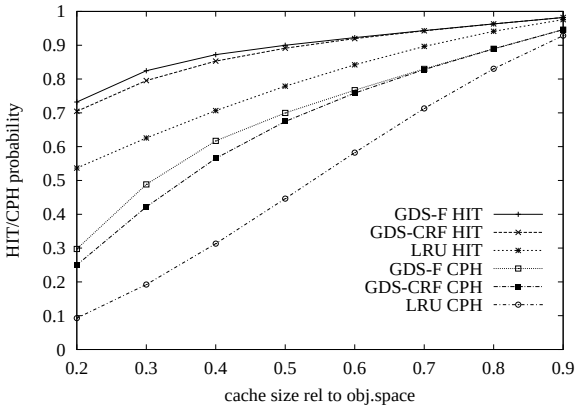

Figure 3: FreqSmall: HIT and CPH ratios.

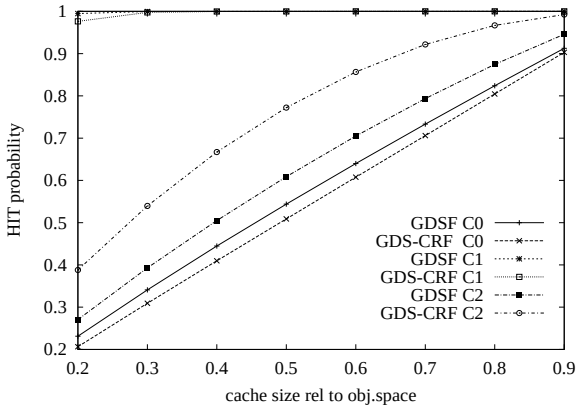

Figure 4: FreqLarge: Class Hits.

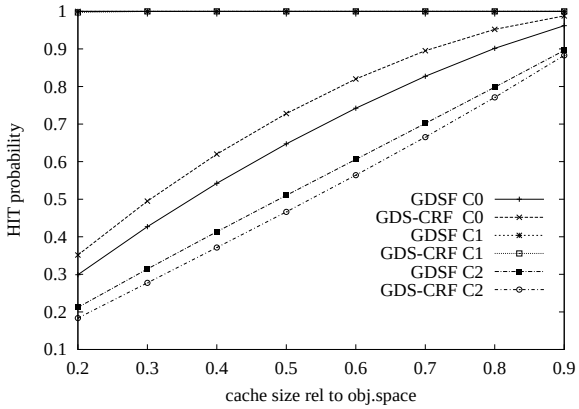

Figure 5: FreqMany: Class Hits.

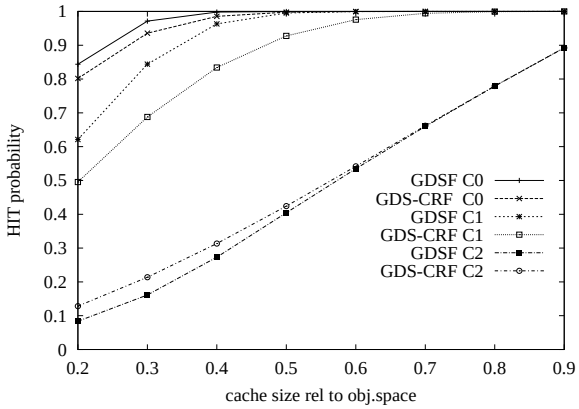

Figure 6: FreqSmall: Class Hits.

nately, we could not find a solution that benefits all configurations.

5.2 Implications of Space Allocation

In order to evaluate the implications of global vs. partitioned space allocation in a fragment-based cache, we consider two partitioning policies, namely:

- **MAXTP,** in which each class is assigned space proportional to the maximum access probability of the templates that include it;
- **CDF,** in which each class is assigned space such that the expected cumulated access probability of stored objects is equal across all classes. For a class, the expected access probability is computed by summing the probabilities of the first k objects in rank order, where $k = \frac{allocation}{avg.size}$.

For both policies, if a class is entitled to more than its maximum needs (i.e., $obj.count \cdot avg.size$), the extra space is reassigned to the other classes based on the same policy. We assume that the space allocation is decided offline, and is fixed for the duration of an experiment.

In the following experiments, the replacement policy for all cache partitions, either global or per-class, is GDS-F. Figure 7 presents the hit ratios for the three configurations when using a global allocation ('GDSF') and when using the two per-class allocations ('Class MAXTP' and 'Class CDF').

The plots illustrate that per-class allocation enables noticeable hit rate improvements for FreqMany throughout the range of cache sizes. For FreqSmall, improvement is observed only for small caches. Comparing the two per-class allocations, CDF provides better hit ratios. The difference is due to changes in per-class hit ratios. Figure 8 illustrates the hit ratios for FreqMany showing that CDF increases more significantly the hit ratio for C0 by reducing the hit ratio for C2.

For the FreqMany and FreqSmall, experiments with better hit ratios exhibited also better CPH ratios. However, the results for FreqLarge (see Figure 9) at small cache sizes illustrate that a better complete-page hit ratio is not always a consequence of a better object hit ratio.

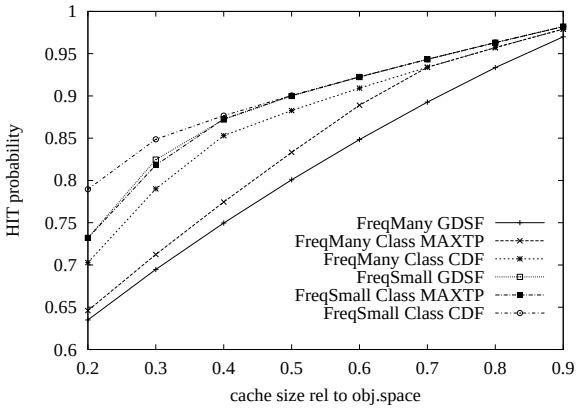

Figure 7: Space Allocation: FreqMany and FreqSmall Hits.

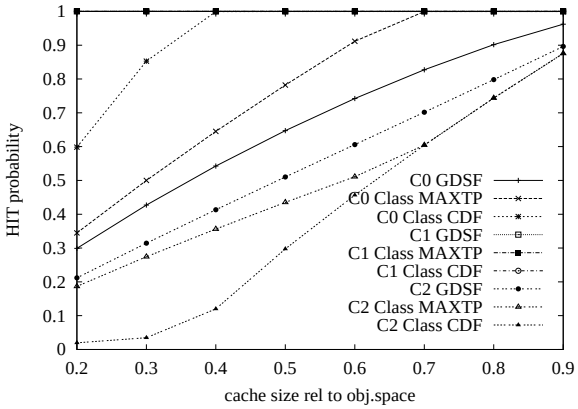

Figure 8: Space Allocation: FreqMany Class Hits.

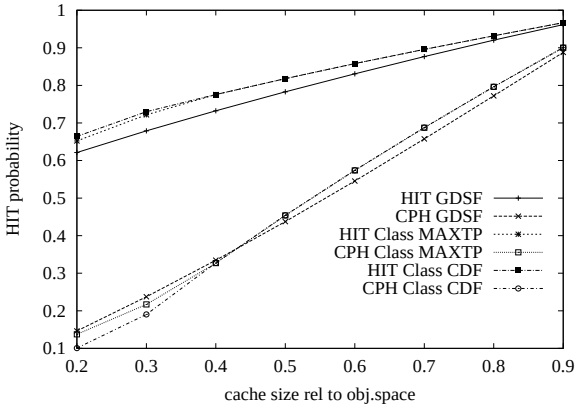

Figure 9: Space Allocation: FreqLarge Hits and CPH.

To summarize, our experiments illustrate that cache partitioning policies that account for fragment-class characteristics can achieve better object and complete-page hit ratio than global cache allocation.

6 Conclusion and Future Work

This paper makes the following contributions to the problem of modeling and caching fragment-based Web content:

- the *Fragment Class* model for fragment-based dynamic content, which captures correlations among content fragments that derive from the structure of dynamic pages;
- the Greedy-Dual Size-Class Relative Frequency, a cache replacement policy that considers the fragment correlations derived from the structure of dynamic pages. The study demonstrates that this policy can provide better hit ratios than optimal cache replacement policies for static content, like GDS-F [1];
- the approach of cache space partitioning based on fragment-class characteristics. This study demonstrates that this approach can provide better object and complete-page hit ratios than the traditional sharing of the entire cache space among all fragment classes.

Albeit small, the performance improvements illustrated in this study advocate the need for further investigation.

Our experiments illustrate that the relative performance of various cache replacement policies depends on content characteristics. Therefore, we submit that an important future-work item is to be able to identify which is the most appropriate cache management policy for given content characteristics. We plan to extend this study with dynamic content that has simple, easy to control characteristics, like in this study, but also with more complex, real-life content.

References

[1] M. Arlitt, L. Cherkasova, J. Dilley, R. Friedrich, and T. Jin. Evaluating Content Management Techniques for Web Proxy Caches. *International Web Caching and Content Delivery Workshop*, 1999.

[2] P. Barford and M. Crovella. Generating Representative eb Workloads for Network and Server Performance Evaluation. *Performance '98/ACM SIGMETRICS '98 and BUCS-TR-1997-006*, 1998.

[3] M. Busari and C. Williamson. On the Sensitivity of Web Proxy Cache Performance to Workload Characteristics. *IEEE INFOCOM*, 2001.

[4] P. Cao and S. Irani. Cost-Aware WWW Proxy Caching Algorithms. *USENIX Symposium on Internet Technologies and Systems*, 1997.

[5] P. Cao, J. Zhang, and K. Beach. Active Cache: Caching Dynamic Contents on the Web. *IFIP International Conference on Distributed Systems Platforms and Open Distributed Processing*, 1998.

[6] J. Challenger, A. Iyengar, K. Witting, C. Ferstat, and P. Reed. A Publishing System for Efficiently Creating Dynamic Web Content. *IEEE INFOCOM*, 2000.

[7] O. Corp. and I. Akamai Technologies. ESI - Accelerating E-Business Applications. *http://www.esi.org/index.html*, 2001.

[8] A. Datta, K. Dutta, H. Thomas, and D. VanderMeer. A Comparative Study of Alternative Middle Tier Caching Solutions to Support Dynamic Web Content Acceleration. *VLDB*, 2001.

[9] A. Datta, K. Dutta, H. Thomas, D. VanderMeer, Suresha, and K. Ramanritham. Proxy-Based Acceleration of Dynamically Generated Content on the World Wide Web: An Approach and Implementation. *SIGMOD*, 2002.

[10] F. Douglis, A. Haro, and M. Rabinovich. HPP: HTML macro-Preprocessing to Support Dynamic Document Caching. *USENIX Symposium on Internetworking Technologies and Systems*, Dec., 1997.

[11] A. Iyengar and J. Challenger. Improving Web Server Performance by Caching Dynamic Data. *USENIX Symposium on Internet Technologies and Systems (USITS '97)*, 1997.

[12] S. Jin and A. Bestavros. Popularity-Aware GreedyDual-Size Web Proxy Caching Algorithm. *International Conference on Distributed Computing Systems 2000*, 2000.

[13] K. Kant and P. Mohapatra. Current Research Trends in Internet Servers. *Performance and Architectures of Web Servers*, 2001.

[14] T. Kelly, Y. M. Chan, S. Janim, and J. MacKie-Mason. Biased Replacement Policies for Web Caches: Differential Quality-of-Service and Aggregate User Value. *International Web Caching Workshop*, 1999.

[15] Y. Lu, A. Saxena, and T. Abdelzaher. Differentiated Caching Services; A Control-Theoretical Ap-

proach. *International Conference on Distributed Computing Systems*, Apr., 2001.

[16] Q. Luo and J. F. Naughton. Form-based proxy Caching for Database-Backed Web Sites. *VLDB*, 2001.

[17] M. Mikhailov and C. Wills. Change and Relationship-Driven Content Caching, Distribution, and Assembly. *Technical Report WPI-CS-TR-01-03, Computer Science Department, Worcester Polytechnic Institute*, Mar., 2001.

[18] M. Mikhailov and C. Wills. Exploiting Object Relationships for Determininstic Web Object Management. *International Web Caching and Content Delivery Workshop*, May, 2002.

[19] P. Mohapatra and H. Chen. WebGraph: A Framework for Managing and Improving Performance of Dynamic Web Content. *Special Issue of Internet Proxy Servers in the IEEE Journal of Selected Areas in Communications*, Sept., 2002.

[20] M. Rabinovich, Z. Xiao, F. Douglis, and C. Kalmanek. Moving Edge Side Includes to the Real Edge – the Clients. *USENIX Symposium on Internet Technologies and Systems*, 2003.

[21] W. Shi, E. Collins, and V. Karamcheti. Modeling Object Characteristics of Dynamic Web Content. *IEEE Global Internet Conference*, Nov, 2002.

[22] W. Shi, R. Wright, E. Collins, and V. Karamcheti. Workload Characterization of a Personalized Web Site – And Its Implications for Dynamic Content Caching. *International Web Caching and Content Delivery Workshop*, May, 2002.

[23] C. Wills and M. Mikhailov. Studying the Impact of More Complete Server Information on Web Caching. *International Web Caching and Content Delivery Workshop*, May, 2000.

[24] H. Zhu and T. Yang. Class-based Cache Management for Dynamic Web Content. *IEEE INFOCOM*, 2001.