

# Subscription-enhanced Content Delivery

Mao Chen

Jaswinder Pal Singh

Andrea LaPaugh

Department of Computer Science  
Princeton University  
Princeton, NJ 08544  
USA  
{maoch, jps, aslp}@cs.princeton.edu

## Abstract

In existing content delivery systems user accesses are popularly used for predicting the request pattern of contents. In novel web applications such as publish/subscribe services, users explicitly provide statements of interest in the form of subscriptions. These subscriptions provide another source of user information in addition to access patterns. This paper addresses the content delivery problem when user-stated interest is available. Each request by a user is either based on a notification about the availability of content that matches the user's subscriptions, or general browsing that is not based on the publish/subscribe service. We propose two approaches to content delivery that exploit both proactive push-time placement and passive access-time replacement based on the subscription information, the access pattern of subscribers, and that of non-subscribers. In our simulation-based evaluation, the two approaches are compared to an access-based caching only algorithm and to three approaches that were proposed for pure notification-driven accesses in our earlier study [5]. The results demonstrate that incorporating subscription information judiciously can substantially reduce the response time, even when only a small portion of accesses is driven by notifications and the subscription information does not reflect subscribers' accesses perfectly. To our knowledge, this work is the first effort to investigate general content delivery and caching enhanced by using subscription information.

## Keywords

Content delivery, caching, subscriptions, access pattern, notification-driven access, placement algorithm

## 1. Introduction

The information needs of content consumers form the key to intelligent caching and content delivery over the Internet. Existing content delivery systems typically determine these information needs based on access patterns. However, many content-intensive web services are based on users' subscriptions, which are statements of interest. This explicit user interest can therefore also be used as a basis for content delivery. Little exploration of subscription-enhanced content delivery has been done.

This paper considers an application scenario in which users may request contents either based on notifications that match their subscriptions or as part of general browsing that is not based on the publish/subscribe service. An example is news delivery. Many news sites such as CNN provide notification service that is based on matching the contents to users' subscriptions. Users may request news pages upon receiving the notifications or browse news sites independent of the notification service. The two types of accesses are called *notification-driven accesses* and *non-notification-driven accesses* or *general browsing* in this paper. In addition, a user may not read all the pages that match the user's subscriptions (and of which they therefore receive notifications).

Our goal is reducing the response time to end-users regardless of the type of access they make, by placing the contents in edge servers that are deployed close to end-users. Each server connects to a group of users, aggregates users' subscriptions, monitors users' accesses, and serves as the cache that is consulted when the requested contents are not in the client caches.

“Pushing to the edge” is a philosophy taken by many content delivery systems [1]. Our system has three important features. First, each content server performs autonomous placement based only on local information at the server. Therefore, our methodology is scalable when content production and consumption is highly dynamic and globally distributed. Second, our approach seamlessly combines proactive publisher-initiated placement and passive client-side caching. Third, this work addresses the storage management at edge servers based on two complementary information sources: the subscriptions from subscribers and the access patterns of all users.

Subscriptions form the basis for estimating future requests from subscribers. This estimation can be done at content generation time by matching the content to subscriptions from all users, which creates an opportunity for early content distribution (the matching of content to subscriptions may be done in an intermediary publish-subscribe service that sits between the user and the content source, and may be based on the entire content or subsets of the content or metadata). The proactive placement benefits all readers of the content, whether they explicitly provided their interest in advance or not.

On the other hand, subscription information by itself is insufficient for designing efficient content delivery algorithms. This is because subscriptions just imply the static distribution of subscribers’ requests. Moreover, it is unrealistic to assume a subscriber will read every page that matches the user’s subscriptions, so subscriptions may not even reflect the request distribution of subscribers perfectly. Therefore, the access patterns of users provide knowledge that is complementary to subscriptions for estimating the actual temporal and frequency request patterns.

We developed two approaches that exploit the type of access, access information, and subscription information. One approach combines all the information sources into a single replacement algorithm for managing the caches, while another one divides a server cache into two portions and processes two types of accesses using different algorithms. Our earlier study [5] presented a set of content distribution approaches for the case where all accesses are notification-driven (i.e. general browsing accesses were not considered in that study). This study determines which of the new

approaches and adapted versions of the old approaches is superior for the new scenario.

As no real-world workload is available to us for this publish/subscribe-based application model, our evaluation study is simulation-based. We extend the simulator and workloads that are built in our earlier study based on the measured behaviour of a busy news site [22]. The new simulator and the workload model the relative composition of two types of accesses and the probability for a subscriber to read the pages that match their subscriptions. Compared to a caching approach based only on observed access patterns, all of our approaches are able to reduce the miss ratio substantially. Interestingly, this is true even when only a small portion of accesses is notification-driven and the probability for subscribers to follow notifications is low.

The contributions of this paper are as follow:

- Presenting the first study for enhancing general content delivery by taking advantage of publish-subscribe information when it is available;
- Proposing several content delivery mechanisms based on access type, subscription distribution, and access pattern, and comparing them under different scenarios to determine the best one;
- Experimentally demonstrating the importance of subscription information in reducing the miss ratio in local cache servers, even when such information is quite limited;
- Modeling the hybrid request sequence and the subscription quality, and incorporating them into a publish/subscribe workload generator.

The next section presents our content delivery strategies, including the new mechanisms proposed specifically for the scenario of this study as well as methods from our earlier study originally for pure notification-driven accesses but also show promise in this scenario. Section 3 introduces a news delivery simulator and the workload that models the hybrid user access and subscription quality. The simulation results are presented in section 4. Section 5 discusses related work to Internet content delivery. Section 6 draws conclusions and indicates future directions for this work.

## 2. Subscription-based content delivery

There are two obvious opportunities to deliver a page to a proxy-server. When a page is generated, a matching engine can determine that the page matches the subscriptions of some users of a proxy, and the page can be proactively placed in the server's cache. Alternatively, as in conventional caching systems, a page can be placed on the cache misses based on the fact of rather than the prediction or likelihood of users' accesses to a page. Our content placement is value-based. A page is given a value determined from the access pattern, the subscription information, and/or the type of access. A content distribution engine can use one or both placement opportunities, using a combination of the three information sources.

### 2.1 Access-based caching: the baseline approach

Caching algorithms passively place pages in a local server only at cache misses. The value of a page is purely based on the access pattern of the page. We use a replacement algorithm called Greedy-Dual\* (GD\*) [14] as the baseline algorithm. GD\* determines the value of a page,  $V(p)$ , from the access frequency, the access recency, the cost to fetch a page, and the size of the page, as in equation 1:

$$V(p) = L + \left( \frac{f(p) \cdot c(p)}{s(p)} \right)^{1/\beta} \quad \text{- equation 1}$$

Where

$L$ : inflation value to capture the access recency

$f(p)$ : number of accesses on the page

$c(p)$ : cost to fetch a page from the publisher

$s(p)$ : page size

$\beta$ : balance factor of popularity and temporal correlation

In our implementation, the reference count of a page is discarded when the page is evicted. The fetch cost is the network distance from the cache server to the origin publisher. The constant parameter  $\beta$  is set manually according to preliminary experiments.

On a cache hit of page  $p$ , GD\* increases the reference count  $f(p)$  and recalculates  $V(p)$  based on the current inflation value  $L$ . A cache miss results in a placement of the requested page. If there is no room for the page, the old pages in the cache are evicted in the order of the pages' values; the inflation value  $L$  is set to be the value of the last evicted page.

By viewing any type of access equally, GD\* can be used in either pure notification-driven content delivery or in content delivery that allows hybrid notification-driven accesses and general browsing.

### 2.2 Single cache and single replacement method

One category of our content delivery strategies combines subscription information and access information into a single evaluation function that is used in both push-time and access-time placement.

#### 2.2.1 Two approaches in our earlier work

Our earlier study for pure notification-driven access identified two good approaches that combine access and subscription information. The first method replaces the frequency factor in equation 1 by equation 2, assuming a page is valuable if many users' subscriptions match the page and many users have requested the page. This approach is referred to as Subscription-GD\*-1 (SG1).

$$f(p) = s + a \quad \text{- equation 2}$$

Where

$s$ : the number of subscriptions matching page  $p$

$a$ : the number of accesses of page  $p$

Another approach, called Subscription-GD\*-2 (SG2), takes into account the relation between the subscriptions and the accesses. If every user requests a page within that user's interest exactly once, the difference between the number of subscriptions matching a page and that of the accesses to the page is the number of requests of the page in future. SG2 replaces the frequency factor in equation 1 by equation 3.

$$f(p) = s - a \quad \text{- equation 3}$$

Where

$s$ : the number of subscriptions matching page  $p$

$a$ : the number of accesses of page  $p$

Both SG1 and SG2 base their decision on whether to store a page  $P$  at a proxy server  $S$  purely on page value. When there is not enough room at  $S$ , all the pages whose value is smaller than that of  $P$  are candidates for eviction. If the total size of all the candidates is smaller than the size of  $P$ , the placement algorithm aborts storing  $P$  at  $S$ . On a cache miss, failing to store a requested page means that the proxy server just fetches the page from the origin site and forwards it to the user without storing the page in the server's cache.

SG1 is directly applicable to the application scenario of this paper, while SG2 needs adaptation. Equation 3 assumes the number of subscriptions that match a page is always larger than the number of accesses of the page up to any point in time. The assumption holds when all requests are driven by notifications. However, if only some accesses are notification-driven as assumed in this study, the result of equation 3 may be negative, which is invalid in the evaluation function 1.

A simple way to deal with the invalid case is to set all negative results to 0 when using equation 3. In this way, when the number of accesses exceeds that of the subscriptions, SG2 ignores the frequency information and evaluates a page purely based on access recency. To distinguish from SG2, the above adapted version of SG2 is referred to as Restrictive-Subscription-GD\*-2 or RSG2 in this paper.

### 2.2.2 First new approach: HUG

Our first new approach in this paper applies the same placement algorithm at both push-time and access-time, as do SG1 and RSG2. Different from SG1 and RSG2, the evaluation function in the new approach distinguishes the information associated with two types of accesses. The new approach is called Hybrid-User-GD\* (HUG) in this paper, since it incorporates the type of access into the GD\* framework using equation 1.

HUG analyzes the frequency information of two types of accesses separately. For notification-driven accesses, the difference between the number of subscriptions matching a page and that of notification-based accesses up to the current time implies the amount of notification-driven accesses in the future. In contrast, the number of non-notification-based accesses in the future is derived from that in the past, according to the traditional LFU algorithms. The following equation estimates the number of future requests of a page using the sum of two factors each of which is associated with one type of access. The evaluation function of HUG is equation 1 after substituting  $f(p)$  with equation 4.

$$f(p) = a_{NS} + (s - a_S) \quad \text{- equation 4}$$

Where

$a_{NS}$  : number of accesses of page  $p$  from non - subscribers

$s$  : number of subscriptions matching page  $p$

$a_S$  : number of accesses of page  $p$  from subscribers

Equation 4 interprets the access frequency of two types of accesses in opposite ways: a high volume of past notification-driven accesses indicates low leftover value of a page for subscribers, while heavy general browsing implies a strong interest of web surfers in the page.

## 2.3 Dual-caches approaches

### 2.3.1 Previous dual-caches approach

Our earlier study presents a Dual-Caches approach that divides a server cache into two portions and uses one portion at push-time and another portion at access-time placement independently. The approach is referred to as *DC* in this paper.

The push-time module of DC uses equation 5 to evaluate pages and decides whether to store a newly published page in the local server using the same replacement policy as SG1 and SG2. The access-time portion is managed using GD\*.

$$V(p) = \frac{f_S(p) \cdot c(p)}{s(p)} \quad \text{- equation 5}$$

Where

$f_S(p)$ : the number of subscriptions matching the content of page  $p$

$c(p)$  and  $s(p)$  have the same meaning as in the equation 1

DC adaptively re-partitions a server's cache according to the publishing pattern and the access pattern. When a page in the push portion (*PP*) is requested, the storage of the page is assigned to the access portion (*AP*). When the push module fails to store a new page and some pages in *AP* have not been requested since the last replacement in *AP*, the storage of the pages is assigned to *PP*. To avoid the imbalance in the partition, the fraction of the storage assigned to each portion is bounded.

In an environment with hybrid user accesses, DC still works if it does not distinguish between notification-driven accesses and general browsing.

### 2.3.2 Second new approach: DC-TA

The key idea of the dual-caches approaches is exploiting different information sources in separate cache portions. DC analyzes the subscription information only at push-time and the access pattern only at access-time, hence DC divides a cache based on the type of operations, pushing and caching.

The access type is also a dimension to partition a cache space. Our second new approach divides a

server’s cache into two equal portions that serve notification-driven and non-notification-driven accesses respectively. In our presentation this approach is called Dual-Caches partitioned by Type of Access (DC-TA).

Using DC-TA, the cache portion that is consulted for notification-driven accesses is called the notification-based portion (*NP*). The other portion that serves general browsing is referred to as the browsing portion (*BP*). NP is managed using SG2 since it is one of the best approaches for pure notification-driven accesses according to our earlier study. BP works as a traditional cache using GD\*. The page evaluation in NP is based on the subscription information and the accesses from subscribers, while the evaluation in BP relies only on the access pattern of general browsing.

NP and BP cooperate in locating requests of end users. If BP holds a page that is requested in a notification-driven access, the locating algorithm returns the page from BP. Similarly, a non-notification-driven request can be satisfied in NP. The cooperative locating between NP and BP is carried out in background and thus is transparent to end-users. From the point of view of users, the two portions seamlessly form a cache as a whole.

A push-time placement for page  $p$  in NP is always before any request of  $p$ , hence  $p$  cannot be in BP yet. The access-time fetching in either portion occurs only when the page is not found in both BP and NP. Therefore, the cooperative locating in the two portions guarantees that no page resides in both portions at the same time.

However, NP and BP do not have to exchange knowledge about the two types of accesses beyond the existence of pages in each portion. In other words, each portion is read-only for the other portion. This design is suitable for an integrated caching system shared by multiple independent service providers. For example, one portion of a server’s cache is assigned to a content delivery service provider such as Akamai [1], and another portion is licensed to a publish/subscribe service provider. The locating engine in DC-TA aggregates only the location information of pages in all the portions, but leaves other privacy-sensitive information to each portion.

### 3. Simulator and workload

#### 3.1 Simulator and workload for news delivery

Because of the difficulty of acquiring real-world data about the applications considered in this paper, our evaluation study is simulation-based. Our simulator mimics news delivery based on analyses and observations about the real-world data in the literature. The simulator assumes a single publisher as a news site and a content delivery system consisting of 100 globally distributed proxy-servers. The network topology of the proxy servers and the publisher is a random graph built using GT-ITM [12]. Each proxy-server is assumed to be deployed close to a set of end-users. The workload includes three traces of a 7-day simulation.

The first trace is a publishing sequence. The publisher generates 30,147 distinct pages in total within 7 days. The second trace is an access trace that includes around 195,000 requests. This trace records the requested page, the issuing time and the closest proxy-server of each request. The popularity distribution of requests follows Zipf-law<sup>1</sup> with  $\alpha$  as 1.5 according to the observation about news requests in [22]. More details about the methods for building the publishing and the access traces are discussed in [5]. For the validation purpose of this study, the access sequence is assumed to consist of two types of accesses.

The third trace is the subscription sequence of end-users. The subscriptions are assumed to be static information that is known in advance. This assumption is reasonable since the update rate of user subscription is usually much lower than the rate of user access. Given the above assumption, the only subscription information of interest is the number of subscriptions matching each page. This information can be built from the access sequence.

#### 3.2 Generating subscription information

Equation 6 determines the number of subscriptions that match a page  $i$  at a server  $j$  based on the total number of requests to  $i$  at  $j$  and the correlation between the subscriptions and the accesses. The correlation is determined by two factors: the ratio of the number of notification-driven accesses to that of all accesses for the page at the server, and the ratio of the number of the notification-driven accesses to

---

<sup>1</sup>  $R_i = 1/i^\alpha$ , the request rate on the page with rank  $i$

that of the subscriptions for a page. The first parameter models the composition of two types of accesses, while the second one quantifies the accuracy of subscriptions or subscription quality.

$$SF_{i,j} = \frac{P_{i,j} \cdot F_{i,j}}{SQ_{i,j}} \quad \text{- equation 6}$$

Where

$SF_{i,j}$  : number of subscriptions matching page  $i$  at server  $j$

$P_{i,j}$  : number of requests to page  $i$  at server  $j$

$F_{i,j}$  : fraction of notification - driven requests to page  $i$  at server  $j$

$SQ_{i,j}$  : probability for subscribers to read page  $i$  at server  $j$

The global parameter  $F$  is the estimate of the fraction of notification-driven accesses for all page/server pairs. The global parameter  $SQ$  stands for global subscription quality, and it is the estimate of the probability for a subscriber to access a page that matches the user's stated interest. Given a distribution and a global parameter  $F$  or  $SQ$ , one can generate  $\{F_{i,j}\}$  and  $\{SQ_{i,j}\}$  for all pages and servers, and hence build the subscription information using equation 6.

Given the lack of evidence in literature about the distributions of  $\{F_{i,j}\}$  and  $\{SQ_{i,j}\}$  as defined, we explore three different distributions to model  $\{F_{i,j}\}$  and  $\{SQ_{i,j}\}$ . The first distribution is a step-wise function with the disjoint point at the estimate value of the sequence. The second distribution is a uniform distribution. The last one is a Gaussian-like distribution. All three distributions have a domain of  $[0, 1]$ . Since the experiments using the three distributions yield similar results, we focus our discussion on the results using a Gaussian-like distribution to model  $\{F_{i,j}\}$  and  $\{SQ_{i,j}\}$ .

### 3.3 Tagging accesses as notification-driven or not

Since HUG and DC-TA require knowledge about the type of access, each access in our request streams should be identified as either notification-driven or non-notification-driven.

Given  $P_{i,j}$  and  $F_{i,j}$  as defined in equation 6,  $P_{i,j} \cdot F_{i,j}$  accesses are randomly chosen from the access sequence to page  $i$  at server  $j$  using a uniform random number generator. The chosen accesses are tagged as notification-driven, while others as general browsing.

## 4. Experimental results

### 4.1 Metric and experimental setup

The publishing site usually resides in a different backbone network; then the communication latency between users and a local proxy-server is usually much smaller than that between users and the publisher. Consequently, a low miss ratio in local servers can be translated into a small response time. We use *global miss ratio* ( $M$ ) on the 100 servers to evaluate the performance of an approach using the following definition:

$$M = \frac{\sum_{i=1}^{100} M_i}{\sum_{i=1}^{100} R_i} \quad \text{- equation 7}$$

Where

$M_i$  : number of misses on server  $i$

$R_i$  : number of requests on server  $i$

To investigate the importance of subscriptions in content delivery systems, the performance of any of our subscription-based approaches is expressed as the relative reduction in  $M$  as compared to the baseline caching-only approach GD\*, as defined in equation 8. A positive value of  $I$  indicates an improvement of a method over GD\*.

$$I_i = \frac{M_{GD^*} - M_i}{M_{GD^*}} \cdot 100\% \quad \text{- equation 8}$$

Where

$I_i$  : the relative improvement of method  $i$

$M_i$  : global miss ratio of method  $i$

$M_{GD^*}$  : global miss ratio of GD\*

In our simulation, the storage capacity of a cache is set to be 5% of the total number of unique bytes requested at the server in the whole simulation. The performances of the approaches are tested under different user models that are characterized by the two global parameters,  $F$  and  $SQ$ .

### 4.2 Comparing RSG2 and HUG

Our earlier study experimentally identifies SG2 as one of the best content delivery and caching approaches we proposed for pure notification-driven accesses. RSG2 and HUG, the two variants of SG2, differ in whether the two types of accesses are

distinguished in the access frequency analysis. Therefore, the two approaches are same when  $F$  is 1, in which case all accesses are notification-driven.

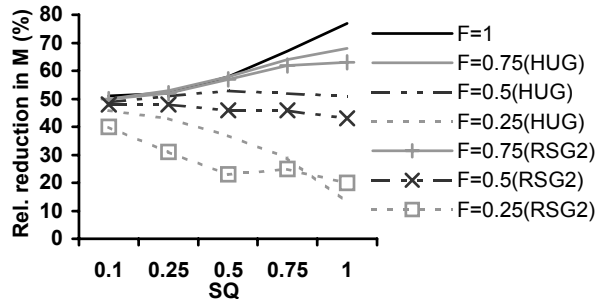
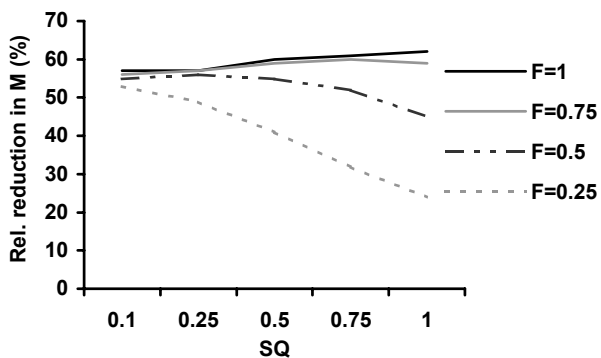


Figure 1. Performances of RSG2 and HUG

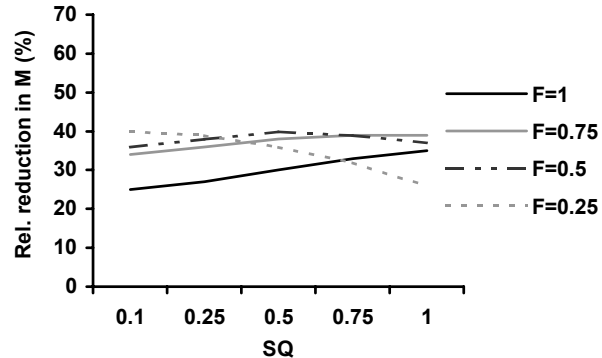
As shown in Fig. 1, RSG2 and HUG reduce the miss ratio over  $GD^*$  in all tested cases. The superiority of HUG to RSG2 in most cases demonstrates the importance of distinguishing access type in evaluating pages.

A larger  $F$  indicates that most accesses are driven by notifications and thus implies more importance of the subscription analysis. The positive correlation between  $F$  and the performances of RSG2 and HUG in Figure 1 exhibits that the both approaches are able to use the subscription information adequately to enhance content delivery.

Interestingly, higher subscription quality itself does not always benefits the performances of the approaches. When  $F$  is 0.25, the performances of RSG2 and HUG degrade with  $SQ$ . A combination of small  $F$  and large  $SQ$  is likely to result in a small number of subscriptions using equation 6, which reduces the importance of subscription information.



(a) Performances of DC



(b) Performance of DC-TA

Figure 2. Comparison of performances of two Dual-Caches approaches

### 4.3 Behavior of DC-TA

#### 4.3.1 Comparing DC and DC-TA

Figure 2 shows the performances of the two dual-caches approaches. Both DC and DC-TA yield better results in terms of miss ratio than  $GD^*$  in all cases. As for RSG2 and HUG, the performances of DC and DC-TA become worse with  $SQ$  when  $F$  is 0.25. As explained before, this is because a combination of small  $F$  and large  $SQ$  results in little subscription information available in the workload.

Under any setting of  $F$  and  $SQ$ , DC yields high gains than DC-TA. Recall that the major distinction between DC and DC-TA is the criterion to partition pages into two cache portions. DC groups all pages that have not been referenced in one portion, and other pages in another portion. In contrast, when a placement is triggered on a cache miss, DC-TA may put the new page into either NP or BP according to the access type. Consequently, DC-TA may evaluate a page using different methods at different reference times of the page. We conjecture that the partition rule based on type of accesses may be one reason for the worse performance of DC-TA.

#### 4.3.2 Influence of cache partition to DC-TA

Different from RSG2, HUG, and DC, the performance of DC-TA does not increase with  $F$ . Given a  $SQ$ , DC-TA achieves the best or the second best result when  $F$  is 0.5. Since DC-TA uses a 50%-50% partition on a server's cache, we conjecture that the good performances of DC-TA when  $F$  is 0.5 are resulted from the suitable cache partition under this  $F$  value. However, a 50%-50% partition may not be good for traces with other  $F$  values.

Recall that DC adaptively updates its partition based on the publishing pattern and the request pattern over time. The intelligent storage partition may be another reason for the superiority of DC to DC-TA in Figure 2. To study the influence of the cache partition, DC-TA is tested under different partitions for each  $F$ . Based on our preliminary experimental results, the optimum storage fraction assigned to notification-based portion are 100%, 100%, 100%, and 75% for  $F$  of 1, 0.75, 0.5 and 0.25 respectively. The optimum fraction value of the notification-driven portion, denoted as OF, is positively correlated but not equal to  $F$ . Interestingly, OF is always larger than  $F$  when  $F$  is less than 1, which means giving higher importance to the notification-driven accesses and subscriptions benefits all users in reducing response time. Particularly, when at least half of the accesses are driven by notifications ( $F \geq 0.5$ ), it is better to use the whole cache for the push-time placement and the access-time placement based on subscriptions and notification-driven accesses only.

DC-TA with Optimum Partition is called ODT in this paper. Figure 3 exhibits a dramatic improvement using the optimum partition over using fixed 50%-50% partition in DC-TA.

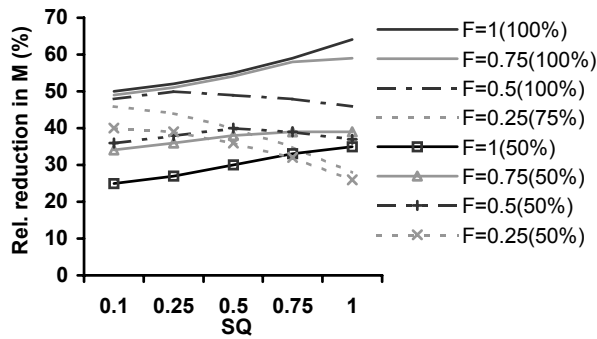


Figure 3. Performances of DC-TA with fixed or optimum partition

Figure 4 compares DC and ODT. Using the optimum partitions, ODT has comparable performances to DC in many cases. When SQ is high, ODT yields the same or even better results than DC. The results in figures 3 and 4 support the hypothesis that adaptive partition based on access pattern is important to dual-caches approaches.

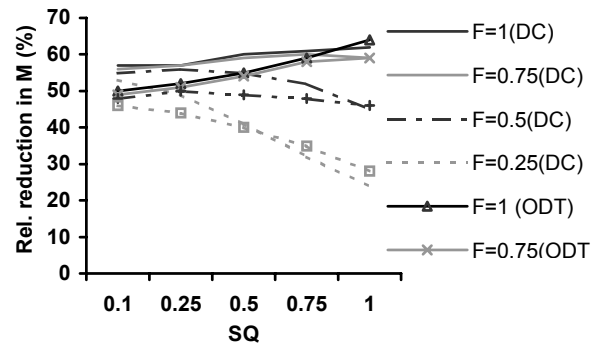
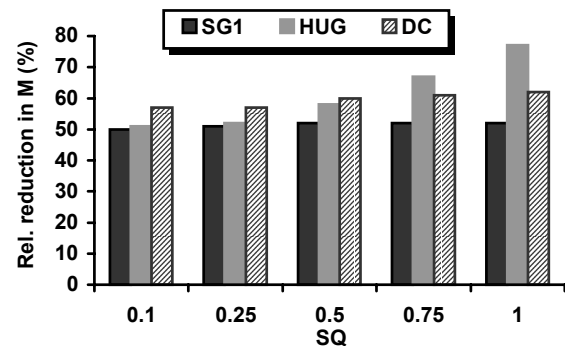


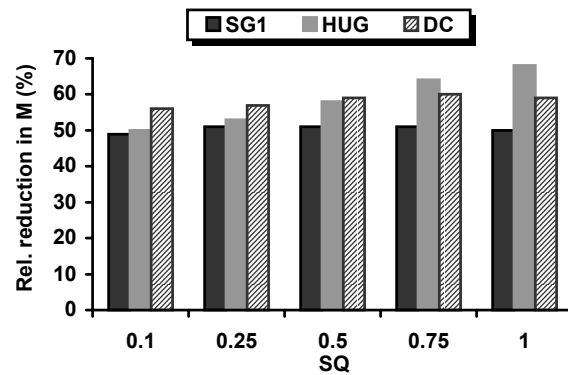
Figure 4. Performances of DC and ODT

#### 4.4 Comparing SG1, HUG and DC

The experiments in sections 4.2 and 4.3 demonstrate that HUG outperforms RSG2, and DC outperforms DC-TA under most settings of  $F$  and SQ. The following figures compare the performances of HUG and DC to that of SG1, one of our previous approaches for pure notification-driven accesses.



(a)  $F = 1$



(b)  $F = 0.75$

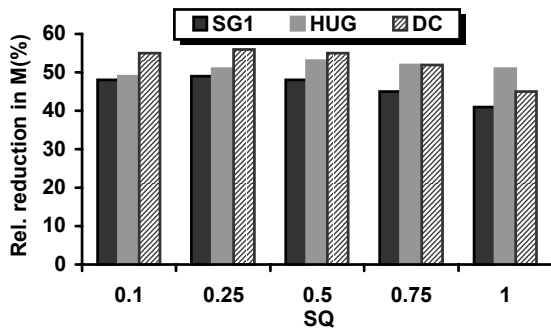
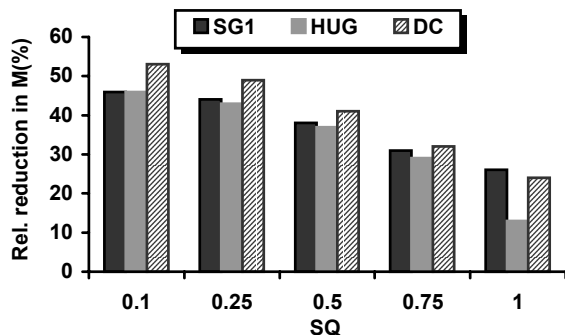
(c)  $F = 0.5$ (d)  $F = 0.25$ 

Figure 5. Performances of SG1, HUG and DC

All of the three approaches reduce the miss ratio over GD\* under all settings of  $F$  and  $SQ$ . Table 1 summarizes the winners for different  $F$  and  $SQ$ .

Table 1. Winners for traces at different  $F$  and  $SQ$ 

$F \backslash SQ$	0.1	0.25	0.5	0.75	1
0.25	DC	DC	DC	DC	SG1
0.5	DC	DC	DC	HUG/DC	HUG
0.75	DC	DC	DC	HUG	HUG
1	DC	DC	DC	HUG	HUG

When both  $F$  and  $SQ$  are high ( $F \geq 0.5$ ,  $SQ > 0.5$ ), HUG yields the highest relative gains over GD\* among the three approaches, hence HUG is best when most requests follow the users' stated interest. DC is the winner for most settings of  $F$  and  $SQ$  when either  $F$  or  $SQ$  is low. As the simplest approach among the three, SG1 yields comparably good results in many cases except for when both  $F$  and  $SQ$  are high. Therefore, SG1 is suitable when computational cost is a major concern.

In application of the proposed content delivery methods, the choice of the three methods should be

made based on the cost of cache misses, the observed user model in terms of  $F$  and  $SQ$ , the affordability of the algorithm complexity, and other application-specific factors.

## 5. Related work

Web caching has been widely applied to distribute content on the Internet. The GreedyDual-Size algorithm combines temporal locality, popularity, fetching cost, and page size into an evaluation function [3]. GreedyDual\* generalizes GDS by balancing the effects of long-term popularity and short-term reference correlation in requests [14].

In contrast to passive caching, prefetching techniques are used to proactively pull information from an original site to a proxy server [7, 9, 26], or from a proxy to a browser cache [10]. The key to prefetching is to predict the pages that will be requested. The prediction is usually based on observed access patterns [10, 26]. The linkage information and/or the content information are also useful in prefetching techniques [4, 9, 20].

Server-initiated multicast achieves proactive content distribution on behalf of publishers. Based on global access information, the popular pages can be automatically pushed to the proxy servers that request the pages frequently [2, 13].

In existing content delivery systems, caching, prefetching, and pushing are mainly based on inferred user interest typically from access patterns. Our work is distinct in exploiting the stated user interest in addition to inferred user interest.

Recently, there has been increasing interest in the placement problem for content delivery networks (CDN). Kangasharju et al. [15] formulate an optimum content replication problem as the problem of minimizing the average number of network hops to fetch a copy, assuming the locating algorithm always gets a copy from the nearest server holding a copy. The optimum placement in an overlay network with a graph topology has been proved to be NP-hard, while there exist polynomial solutions for other topologies such as tree [19, 21]. Cidon et al. [6] present an optimum solution in a distribution tree, but the approach assumes a static environment in which all requests are given precisely in advance and the network condition does not change.

Kangasharju et al. [15] propose four heuristics for content placement, but the best one needs to exploit global knowledge about the network topology, the global reference distribution and the

global content image at different times. Qiu et al. [23] propose several heuristics to choose  $M$  replica sites from  $N$  candidates for a given site, also assuming a relative stable reference pattern at the candidate sites.

In this paper, each cache server is autonomous in managing its storage using local subscription and access information. Therefore, our approaches are scalable in a globally distributed system that faces dynamic content publishing and request patterns, and network conditions.

Karlsson and Mahalingam [17] argue that caching works at least as well as replica placement algorithms, if the cache storage is ample and the replacement algorithm runs periodically rather than after each access. For combining placement and caching, Korupolu and Dahlin propose a “static partition” approach that divides a server’s cache into two portions and runs placement and replacement algorithms on the two portions separately [18]. The approach is analogous to our DC algorithm, but several major distinctions exist between the two. DC takes into account stated user interest rather than being purely based on request distribution. For the placement portion, our push-time placement is triggered by content generation and matching the content to users’ subscriptions, while placement is called periodically in [18]. Moreover, DC re-partitions a server’s cache adaptively to the access and publishing pattern.

An important component in CDN is load balancing. Load balance is usually achieved by request redirection [1, 8] using different hashing schemes [16, 25]. Caching can be used as complimentary to request redirection-based CDN. To the best of our knowledge, the existing CDNs that work with conventional caching systems still rely on request patterns only. Gadde et al. [11] suggest a natural limit to the marginal benefits of redirection-based CDNs, since the hit ratio in proxy caches increases dramatically as ISPs serve larger user communities. A recent experimental study on content delivery systems also points out the importance of widely deployed proxy caches as compared to a separate CDN [24].

## 6. Conclusion

We proposed several content delivery approaches for an environment where a publish-subscribe service exists, and user may request contents based on notifications that satisfy their subscriptions or by

general browsing. In addition to three approaches from our earlier study that assumed all accesses are based purely on notifications (with necessary adaptation to the new scenario), we presented two new methods that exploit information about access type, access patterns, and subscriptions. Our performance metric is the miss rate in local caching servers, assuming a lower miss rate can generally be translated into a smaller response time.

We find that all of our approaches that use subscription information outperform an access-based caching-only algorithm. Interestingly, this is true even when only a small fraction of accesses is notification-driven and when subscriptions do not predict subscribers’ accesses very well. Especially, when users’ stated interest matches users’ access patterns fairly well, HUG, the new approach considering the access type, is the best approach; otherwise, our previous dual-caches approach DC outperforms our other approaches. As the best algorithm in our earlier study on pure notification-driven accesses, SG2 is not the best approach in the new scenario any more, even after necessary adaptation.

A major direction for our future work is to examine the use of subscription information in cooperative placement and locating schemes for a distributed system.

## References

- [1] Akamai. <http://www.akamai.com>.
- [2] Besravros, A. Demand-based Document Dissemination to Reduce Traffic and Balance Load. In Proceedings of the 7<sup>th</sup> IEEE Symposium on Parallel and Distributed Processing (SPDP’95), 1995.
- [3] Cao, P. and Irani, S. Cost-Aware WWW Proxy Caching Algorithms. In Proc. of USENIX Symp. on Internet Technology and Systems, 1997.
- [4] Chi, E. H., Pirolli, P., Chen, K., and Pitkow, J. Using Information Scent to Model User Information Needs and Actions on the Web. CHI 2001, 3(1), 490-497
- [5] Chen, M, LaPaugh, A. and Singh, J. P. Content Distribution for Publish/Subscribe Services. In Proc. of ACM/IFIP/USENIX Middleware 2003.
- [6] Cidon, I., Kuten, S., and Soffer, R. Optimal Allocation of Electronic Content. In Proceedings of IEEE Infocom 2001.

- [7] Deolasee, P., Katkar, A., Panchbudhe, A., Ramamritham, K., and Shenoy, P. Adaptive Push-Pull: Disseminating Dynamic Web Data. In Proceedings of WWW10, 2001.
- [8] Digital Island. <http://www.digitalisland.com/>.
- [9] Duchamp, D. Prefetching Hyperlinks. In Proc. of USENIX Symp. on Internet Technologies and Systems, 1999
- [10] Fan, L., Cao, P., Lin, W., and Jacobson, Q. Web Prefetching Between Low-Bandwidth Clients and Proxies: Potential and Performance. In Proceedings of ACM SIGMETRICS, 1999.
- [11] Gadde, S., Chase, J., and Rabinovich, M. Web Caching and Content Distribution: A View From the Interior. 5<sup>th</sup> International Web Caching and Content Delivery Workshop (WCW '00), 2000.
- [12] GT-ITM: Georgia Tech Internetwork Topology Models. <http://www.cc.gatech.edu/fac/Ellen.Zegura/graphs.html>
- [13] Gwertzman, J. and Seltzer, M. An Analysis of Geographical Push-Caching. 1997.
- [14] Jin, Shudong and Bestavrou, A. GreedyDual\* Web Caching Algorithm: Exploiting the Two Sources of Temporal Locality in Web Request Streams. Computer Comm., 24(2), 174-183, Feb. 2001.
- [15] Kangasharju, J., Roberts, J., and Ross, K. W. Object Replication Strategies in Content Distribution Networks. In Proceedings of WCW'01: Web Caching and Content Distribution Workshop, June 2001.
- [16] Karger, D., Lehman, E., Leighton, F. T., Levine, M., Lewin, D., and Panigrahy, R. Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web. In Proceedings of the 29th Annual ACM Symposium on Theory of Computing, pages 654-663, 1997.
- [17] Karlsson, M. and Mahalingam, M. Do We Need Replica Placement Algorithms in Content Delivery Networks. In Proc. Of WCW '02, 2002.
- [18] Korupolu, M. R. and Dahlin, M. Coordinated Placement and Replacement for Large-Scale Distributed Caches. IEEE Transactions on Knowledge and Data Engineering, 2002 (Vol. 14, No. 6), pp. 1317-1329.
- [19] Krishnan, P., Raz, D., and Shavitt, Y. The cache location problem. IEEE/ACM Transactions on Networking, 8(5): pages 568-582, October 2000
- [20] Lieberman, H. Letizia: An Agent That Assists Web Browsing. Proceedings of the 1995 International Joint Conference on Artificial Intelligent, 1995.
- [21] Li, B., Golin, M. J., Italiano, G. F., and Deng, X. On the Optimal Placement of Web Proxies in the Internet. In Proc. of IEEE INFOCOM'99, 1999.
- [22] Padmanabhan, V. N. and Qiu, L.-L. The Content and Access Dynamics of a Busy Web Site: Findings and Implications. In Proc. of ACM SIGCOMM 2000.
- [23] Qiu, L., Padmanabham, V. N., and Voelker, G. M. On the placement of web server replicas. In Proceedings of 20<sup>th</sup> IEEE INFOCOM, 2001.
- [24] Saroiu, S., Gummadi, K. P., Dunn, R. J., Gribble, S. D., and Levy, H. M. An Analysis of Internet Content Delivery Systems. In Proc. of OSDI '02, 2002.
- [25] Wang, L.-M., Pai, V., and Peterson, L. The Effectiveness of Request Redirection on CDN Robustness. In Proceedings of the Fifth Symposium on Operating Systems Design and Implementation (OSDI'02), 2002.
- [26] Venkataramani, A., Yalagandula, P., Kokku, R., Sharif, S., and Dahlin, M. The Potential Costs and Benefits of Long-term Prefetching for Content Distribution. Technical Report TR-01-13, UT, Austin, 2001.