Hoarding Location-Based Data Using Clustering

Susanne Bürklen, Pedro José Marrón, Kurt Rothermel, and Timo Pfahl
Institute of Parallel and Distributed Systems
Universitaet Stuttgart
70569 Stuttgart, Germany
{buerklen,marron,rothermel}@ipvs.uni-stuttgart.de,timopfahl@gmx.de

ABSTRACT

The proliferation of mobile devices and the fact that high-bandwidth and continuous connectivity is not available everywhere, has led to the creation of hoarding algorithms that attempt to mitigate the problems related with disconnected operation and with the operation in areas where bandwidth is either scarce or expensive. In this paper, we present a hoarding scheme for location-based data in semi-structured information spaces, such as the World Wide Web, which relies on clustering of semantically related data items. We show by means of experimental evaluation that our clustering-based approach outperforms existing hoarding techniques that do not make use of clustering by a factor of more than 2 in terms of hoard cache hit ratio.

Categories and Subject Descriptors: H.3 [Information Storage and Retrieval]: Systems and Software, Online Information Services

General Terms: Algorithms, Management.

Keywords: Location-Based Systems, Hoarding.

1. INTRODUCTION

Looking at current trends in communication patterns, it seems clear that there is an increasing demand for information anytime and anywhere. Mobile users want to be able to access data independently of where they are and where this data is located. In particular, they want to have mobile access to the World Wide Web, which is the major source of information nowadays and the basis of many applications.

Wireless communication technology is a prerequisite for mobile data access. Wireless WANs, such as 2G and 3G cellular networks, provide wireless connectivity virtually everywhere. However, even 3G networks only provide moderate data rates up to a few Mbps, typically 300 to 500Kbps. Moreover, communication channels may be subject to substantial error rates, high latency and frequent disconnections. A complementary technology is based on wireless LAN (WLAN) hotspots, which are isolated "islands" of connectivity. Within a limited range, they provide comparatively cheap or sometimes even free high-speed Internet access and at least an order of magnitude higher data rates than wireless WANs.

The two technologies, WLAN hotspots and wireless WANs can be combined to optimize mobile data access. Mobile devices can use the available connectivity at a given hotspot to prefetch and locally cache data their users might access in future. This approach, called hoarding, aims at minimizing wireless WAN usage in order to decrease cost and access latency, in particular for larger data items. Moreover, it also supports disconnected operation since (a portion of) the data requests can be served from the local hoard cache and by this increases the robustness against disconnections.

In contrast to caching, the goal of hoarding is to fetch a data item into the cache *before* the first access occurs. Therefore, it is necessary to predict which data items will be needed next. This prediction, called hoarding decision, may be done by the user himself, telling the system which items to hoard, or it may be done automatically by the system exploiting different types of information, such as user interests or location-dependency and syntactic/semantic relationships of data. In this paper, we present an automatic hoarding mechanism that assumes requests to data items to be location-dependent. Moreover, the algorithm takes into account the semantic distance [14] of data items, which is derived from the users' navigation behavior in a semi-structured information space.

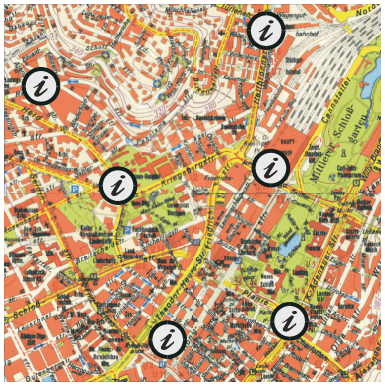


Figure 1: Infostations spread over a city center

For the purpose of this paper, we use the term *infostations* [2, 25] to refer to hotspots that provide hoarding functionality. Fig. 1 depicts the map of a city center including

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
MobiWAC'06, October 2, 2006, Torremolinos, Malaga, Spain.
Copyright 2006 ACM 1-59593-488-X/06/0010 ...$5.00.

infostations positioned at selected points of interest. The circles indicate the areas covered by wireless LAN. Infostations are responsible for selecting the data items to be downloaded into the hoard cache of mobile devices. In most hoarding algorithms described in the literature, the hoarding decision is based on information collected about the data access behavior of users. They mainly differ in the type of data they consider (e.g., unstructured and semi-structured) and the information they collect and exploit for the hoarding decision.

In unstructured information spaces, an infostation usually collects information about the popularity of individual data items. The items that seem to be most popular at a given location are then downloaded to the appropriate clients. In semi-structured information spaces, such as the World Wide Web, we can additionally analyze the behavior of users as they navigate through it. With this analysis we can leverage relationships among data items that can be used to determine their semantic distance. Such a relationship can be a (logical) link, which indicates that the linked items were accessed subsequently in a user session. In addition to the popularity of data items, an infostation can now take the popularity of links into account when making the hoarding decision.

The hoarding scheme proposed in this paper takes into account how users navigate the Web. Moreover, it classifies pages into content and transit pages and uses this classification for clustering. A transit page is one that is (mainly) used to navigate to a content page, which is a page the user is actually interested in. In our scheme, the unit of information used for hoarding is a cluster of Web pages, rather than individual pages. Roughly speaking, a cluster contains at least one content page plus a sequence of transit pages. In addition, our algorithm computes the relevance for each cluster formed by taking into account its access probability and size. Although our hoarding algorithm has been designed for mobile Web access, it is generic enough to be also used for other semi-structured information spaces.

Our approach is based on the assumption that information access is location-dependent, i.e., the information a user is interested in depends on its current location. This assumption is valid for most applications taking into account the users' position, such as navigation systems, location-based services, sentient computing or context-aware information systems. Location-based applications typically "focus" their information needs on a very limited fraction of the entire information space, where this fraction depends on the user's current location. To exploit this location-dependency for the hoarding decision, we monitor the collective user behavior on a per-location basis. In other words, each infostation knows how the users residing in its service area typically access the data. Of course, this knowledge must be dynamically adapted since the access behavior will change over time. The advantage of monitoring the collective access behavior is that those data is selected for hoarding that currently is of interest at a particular location. Further, hoarding also works for users visiting a location for the first time. However, the problem with considering collective behavior only is that the hoarding decision may not be optimal for an individual user. However, this effect diminishes if we exploit user profiles including interests, such as in culture or sports, and monitor the access behavior on a per-location per-interest group basis.

In our experimental evaluations, we compare our hoarding approach to schemes that do not make use of structural information and/or do not perform clustering of semantically related data items. The results show that our hoarding algorithm more than doubles the hoard cache hit ratio compared to existing schemes.

The remainder of this paper is structured as follows. After discussing related work in Section 2, we give a short overview of our hoarding approach in Section 3. In Section 4, we introduce the concept of an information graph, which is used by infostations to represent the access behavior of users. The clustering algorithm and the steps involved in computing the hoarding decision are discussed in Section 5. In Section 6, we discuss the experimental results of our evaluations. Finally, we conclude with a short summary in Section 7.

2. RELATED WORK

To overcome the shortcomings of wireless networks, such as low bandwidth, frequent disconnections, and high latency, various optimization techniques have have been developed for mobile data access.

Caching techniques cache data objects retrieved over the communication network to avoid communication for further accesses to this object (e.g., see [20]). Semantic caching strategies have been developed that exploit geographic locality when replacing objects, e.g., [21]. However, caching aims at optimizing the second and following acceses to an object, while hoarding tries to optimize the first-time access by prefetching the object. As will be seen below, some replacement strategies, such as LRU and LFU, can be applied in hoarding schemes to select the "hot items" to be prefetched (e.g., see [12, 14]).

Several prefetching techniques have been proposed for reducing the user-perceived latency to access Web pages (e.g., see [10], [24]). They predict the set of pages the user probably will access next based on the user's access history. In contrast to that, in our hoarding scheme, the prediction is based on the access behavior of all users residing in a given geographic area.

The Coda file system [11] was one of the first systems that used the concept of hoarding to allow for disconnected operation. However, the hoarding decision requires user interaction. While SPY UTILITY [23] automatically detects logically related files, called working sets, it also requires user interaction to select the sets to be hoarded. SEER [14] hoards files without user interaction. It also performs clustering of related files based on their semantic distance, where files referenced at the same time are assumed to be semantically related. In [13], the authors compared SEER to pure LRU hoarding without clustering. The results revealed that in most cases, LRU hoarding achieved the better results. The authors of [26] propose another automatic file hoarding scheme, where each file is assigned a priority based on access recency, access frequency and the time a file was opened. The file priority and other file-specific parameters determine which files are to be hoarded. All of the above file hoarding schemes are tailored to the characteristics of file access rather than Web access. Hence, they neither model the navigational access behavior nor the characteristics of Web pages (e.g., content or transit pages), which both are important for determining the semantic distance of Web pages. Moreover, these schemes make their hoarding decision on a per-user basis, while our scheme does it on a per-location basis.

In the Map-on-the-Move project [25], drivers are provided with map information at different resolutions depending on the speed they travel through the transmission areas of infostations. However, this approach is tailored to map data.

Approaches such as [15,19] combine the benefits of hoarding and cooperative caching methods. The idea is to increase the hoarding space by letting mobile devices share their hoard caches. This research focuses on distributed cache management rather than the hoarding decision itself.

In [12], the authors propose a hoarding mechanism for location-dependent data, where the hoarding decision is based on how frequently a data item is accessed by all users at a given location. This LFU hoarding scheme was designed for unstructured information spaces and does not analyze the navigational access behavior to derive semantic relationships between data items. Another hoarding scheme for location-dependent data has been proposed in [6]. While its hoarding decision is based on the collective navigational behavior of all users at a given location, it does not provide for clustering semantically related data items.

The hoarding scheme presented in this paper provides for clustering. Several clustering methods have been also proposed in the field of information retrieval, which, however, do not aim at hoarding. The authors of [4] propose a clustering method to group the pages in a result set of a search query based on other queries and their associated result sets. Approaches such as [8,16,18] thematically cluster documents based on common keywords.

3. ASSUMPTIONS AND OVERVIEW

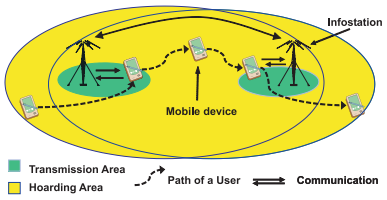


Figure 2: System Model

As depicted in Fig. 2, our system model comprises *infostations* and *mobile devices*. An infostation is a stationary server connected to the Internet and equipped with a WLAN access point. We use the term *transmission area* to refer to the transmission range of the access point and *hoarding area* to the geographic region, the infostation is responsible for. The size of a hoarding area depends on the number of information items per unit of area which are requested in the specific region. The hoarding area is much larger than the transmission area.

Typically, there will be many infostations spread over a region like a town (see Fig. 1). To ensure complete support for hoarding in this region, each hoarding area should overlap with at least one transmission area of another infostation, as depicted in Fig. 2. Doing so, users are supplied with the information they will probably access until they reach the next infostation. Infostations exchange their knowledge about user requests. In particular, they distribute those requests to infostations in whose hoarding area they have been performed.

In the following, we concentrate on the operation of a single infostation.

We assume that a mobile device, such as a PDA or smart phone, is equipped with WLAN capabilities and possibly with wireless WAN connectivity. Mobile devices can be in the following states: *strongly connected*, *weakly connected*, or *disconnected*. A mobile device is strongly connected while it is in the transmission area of an infostation. It is weakly connected if only wireless WAN connectivity is available and in the disconnected state if no connection to a network is possible.

Our approach assumes data requests to be location-dependent, i.e., the probability a certain data item is requested by a user, depends on the user's current location. So, typically only a small fraction of the underlying information space will be "relevant" in an infostation's hoarding area [7]. Moreover, we assume that users are grouped into interest groups, where members of the same group have similar information needs. For the sake of simplicity, we will assume in the following that there is only a single interest group. An extension of our scheme to support different interest groups is straightforward.

Fig. 3 depicts the overall process of our hoarding scheme. A mobile device runs a browser allowing the user to navigate the Web. In addition, it maintains a *hoard cache*, which stores the Web pages downloaded from the most recently visited infostation.

When a user requests a page, the *hoard cache manager* checks whether or not this page is stored in the hoard cache. If the page is cached, it can immediately be returned to the user. In case of a hoard miss, the page may be accessed over the network, provided the device is connected. In either case, a hoard miss is signaled to the mobile device.

A mobile device maintains a *log file*, which is used to keep track of the locally requested Web pages. Whenever a page is requested, a record is added to the log file containing the URL and size of the page, as well as the time and location of the request. In order to determine the location of Web page requests, we assume that each mobile device can determine its position. The location information associated with a request should allow an infostation to determine in which hoarding area the request was issued. Therefore, GPS accuracy, for example, is fully sufficient. Even cell IDs can be used if a hoarding area is defined as a set of cells. Of course, in this case, cellular WAN connectivity is mandatory.

When a mobile device moves into the transmission area of an infostation, it becomes strongly connected and uploads its log file and hoard cache size to the corresponding infostation, which uses the log file to update the corresponding *information graph*. An information graph models the aggregated browsing behavior of all users of an interest group while they are in the hoarding area. In contrast to the dependency graph used in prefetching technologies, the information graph includes information about the frequency of accesses, which pages have been requested including the information whether or not they are content pages, how they relate to each other, and the location of requests.

Due to our simplifying assumption that there exists only a single interest group, we will only consider a single information graph in the following. Actually, with multiple interest groups, we would have to (conceptually) maintain an individual information graph per group.

The *Graph Update Component* divides the uploaded log file into browsing sessions, defined as a sequence of temporally related page requests. Since graph update is performed

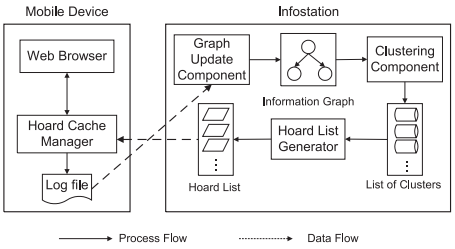


Figure 3: Hoarding Process

each time, a log file is uploaded, our approach permanently adapts to user access behavior.

The classification of pages into content and transit pages is the basis of our *clustering component*. The pages represented in the information graph are grouped into clusters, which are the hoard units to be downloaded to mobile devices. Each cluster includes at least one content page plus a sequence of transit pages needed to navigate to the content pages in the cluster.

The *hoard list generator* assigns a relevance value to each of the clusters that depends on the probability of its pages being accessed and its size. The final step involves the generation of an ordered list of Web pages, which we call a *hoard list*.

As we will see below, clustering and hoard list generation is done only periodically due to performance reasons, while graph update and hoard list download is performed whenever a log file is uploaded.

Let us now discuss each of the algorithms in the infostation in more detail.

4. GRAPH UPDATE

As described in the previous section, the process of updating the information graph involves two steps: (1) the analysis of the log file, which includes the computation of visit periods, the computation of session boundaries and the classification of Web pages, and (2) the actual information graph modification.

4.1 User Log File Analysis

A user log file is composed of log entries which contain the URL of the page and its size, as well as the location and time of the request. We calculate the *visit period* of a page by subtracting the timestamps of two consecutive log entries and use this information to subdivide the log into so-called *sessions*. A session is a subset of consecutive log entries in a log file, where the visit period of each page falls below a threshold value. For our experiments and following the suggestions in [1], we have set this threshold value to 15 minutes.

We also use a page's visit period to detect whether it is a content page or a transit page. Content pages are assumed to be visited considerably longer than transit pages. Moreover, a log file usually contains more transit pages than content pages, since users tend to click through a series of transit pages to find the page they are interested in. Therefore, we decided to use the geometric mean as metric to filter out content pages. Let m_g be the geometric mean of the visit periods of all pages in the log, l.period the visit period of page l.URL requested in log entry l. Then l.URL is classified a content page, if $l.\text{period} * \gamma > m_g$, otherwise, it is considered to be a transit page. Parameter γ determines the selectivity of this content page filter. In our experiments, a value of $\gamma = 0.75$ turned out to be appropriate.

As mentioned above, the (collective) access behavior is tracked on a per-location basis. Since a session models a sequence of logically related accesses, we assign a single location to each session: The location associated with the first entry of a session defines the location of this session. When a log is uploaded to an infostation, it uses all sessions whose location is within its hoarding area to update its information graph (see below) and forwards the sessions to the corresponding infostations. For the sake of simplicity, we will not further detail this inter-infostation collaboration.

4.2 Information Graph Modification

Each infostation maintains an information graph that encodes the information received as part of the user log file into a weighted graph, where the vertices represent the individual Web pages and the edges contain information about the navigational browsing patterns of users.

More formally, the information graph IG is a tuple $IG = (V, E, g, B, z)$, where V is the set of vertices representing Web pages; E is the set of edges; g is the root vertex; $B \subseteq V$ is the set of session start vertices; and z is an accumulative vertex used for bookkeeping of sessions and user revisit patterns.

Additionally, there is extra information encoded in both vertices and edges. Each vertex $v \in V$ has attributes v.URL and v.size, that contain the URL and size of the page associated with it. Furthermore, a vertex also has counters v.request and v.content that record, how many times a page has been requested and has been considered to be a content page, respectively. Analogously, each edge $e \in E$ has a counter e.traversal that keeps track of how many times its source and sink vertices have been requested consecutively over the lifetime of the information graph.

Using this information graph and the uploaded log file, the update algorithm performs the following steps for each session $\mathcal{L}_s = \{l_1, \ldots, l_m\}$ identified during the log analysis step. For the ease of exposition, we assume that a log entry l_i has a corresponding vertex v_i in the information graph.

1. For each entry $l_i \in \mathcal{L}_s$, increment the $v_i.\textit{request}$ counter. If l_i is a content page, increment $v_i.\textit{content}$, too.
2. Increment the traversal counter of an edge e according to the following cases:
   (a) If l_i is a session start page, update edge $e = (g, v_i)$.
   (b) If l_i is a session end page, update edge $e = (v_i, z)$.
   (c) For all other entries l_i with predecessor l_{i-1}, update $e = (v_{i-1}, v_i)$ if l_i appears in the session for the first time. Otherwise, update $e = (v_{i-1}, z)$.

Following this algorithm, the information graph only adapts very slowly to changing user behavior. In fact, the ability to adapt becomes even worse over time, since past and current page accesses are counted equally. Therefore, we integrate an *ageing mechanism*, which divides time into epochs. Whenever an epoch ends, for each of the aforementioned counters, we compute a smoothed value using an exponentially weighted moving average function.

Therefore, for each counter (*v.request, v.content, e.traversal*), we additionally maintain an auxiliary counter which takes care of recording the number of requests, traversals, etc. in the current epoch. When an epoch ends, we compute the value of the corresponding counter as follows and then reset the auxiliary counter:

$$\text{counter} = \delta \cdot \text{counter} + (1 - \delta) \cdot \text{aux_counter}$$

where δ is a smoothing factor. The actual implementation of our algorithm integrates this ageing mechanism with the cluster traversal algorithm described below so that no extra graph traversal is required. In our experiments, the best results were obtained for $\delta = 0.5$.

5. CLUSTERING AND HOARD LIST GENERATION

At the end of an epoch, the infostation traverses the information graph and builds a list of *information clusters* sorted by relevance to be used as hoard units.

A cluster is defined as a subset of the information graph that contains a sequence of vertices lying on a path from the root vertex g to one or more content vertices. For each content vertex, we build a separate cluster and, hence, clusters might overlap.

Our clustering algorithm is based on a modified version of the Bounded Depth First Search (B-DFS) algorithm. It extends B-DFS in two aspects. First, vertices may be visited multiple times to potentially traverse all paths in the information graph. Secondly, the traversal is bounded in terms of the access probabilities associated with the individual paths.

Before describing our algorithm, we have to introduce the terms edge probability, path probability, and content probability. Let $e = (u, v)$ be the edge connecting vertices u and v. The edge probability $p(e)$ represents the probability that a user will request v.URL immediately after u.URL. More precisely, the metric is defined as follows. Let $E_{\text{out}(u)}$ be the set of edges outgoing from u. Then,

$$p(e) = \frac{e.\text{traversal}}{\sum_{e' \in E_{\text{out}(u)}} e'.\text{traversal}} \quad (1)$$

We define the path probability $p(\text{path}_v)$ from vertex $v \in V$ to the root vertex g as the product of the probability of each individual edge that forms the path:

$$p(\text{path}_v) = \prod_{e \in \text{path}_v} p(e)$$

Analogously, we define the content probability of a vertex as the ratio between its content and request counter, as follows:

$$p(\text{content}_v) = \frac{v.\text{content}}{v.\text{request}}$$

As mentioned above, our clustering algorithm potentially traverses all paths. In order to avoid cycles, each vertex maintains a mark which is set the first time, this vertex is visited in a root-to-leaf direction. The mark is removed during backtracking, which is initiated in three cases: (1) The probability of the path to the following vertex is smaller than a threshold value. (2) The accumulative vertex z is reached. In this case, a session ended or a user revisited pages, respectively. (3) A marked vertex is reached. Note that the *path probability threshold* is an important parameter for limiting the computation overhead.

At the end of this step, we have a set of clusters with statistical information such as their size and access probability.

The *hoard list generator* sorts the clusters according to their *relevance*. Since the aim of hoarding is to maximize the number of cache hits when a mobile user requests Web pages and the hoard cache is limited, we consider a cluster's access probability per byte rather than per page when computing the relevance. Given a cluster C and content vertex v, that was responsible for building it, we define its relevance $r(C)$ as:

$$r(C) = \frac{(p(\text{path}_v)) \cdot (p(\text{content}_v))}{C.\text{size}}$$

where $C.size$ is the sum of the sizes of each individual vertex contained in C. The final step involves the generation of the hoard list using the ranked clusters found by the previous algorithms and the hoard cache size reported by the client.

6. EVALUATION

6.1 Methodology and Experimental Setup

From a user's perspective, the major performance parameter of a hoarding scheme is the so-called content hit ratio, which is the probability of a content page requested by a user being found in the hoard cache. Remember, that transit pages are only used to navigate to content pages. Let R be the set of requested content pages and H the set of pages in the hoard cache. Then the content hit ratio is defined as $\frac{|H \cap R|}{|R|}$.

Another important parameter is energy consumption. Our analysis showed that the proposed hoarding scheme can also be used to save energy, assuming current wireless LAN and WAN technologies. Knowing the content hit ratio that can be achieved for different hoard cache sizes, we can determine the cache size for downloading that is optimal in terms of energy savings. For the scenarios described below, we could achieve energy saving of more than 40%. However, a detailed discussion of energy aspects is beyond the scope of this paper.

Even for location-based applications, a fraction of the issued data requests may not be location-based. In our experiments, we will only consider the portion of the location-based requests. Taking into account the other requests as well is not necessary, since each of these requests will most probably lead to a hoard cache miss. So the hoarding gains discussed below only hold for the fraction of location-based requests. For the remaining fraction, other optimization techniques, such as caching, must be applied, which, however, are not subject of this paper. Consequently, the larger the fraction of location-based requests, the higher is the improvement achieved by hoarding.

In our experiments, we assume that each infostation is associated with a so-called location-based subspace. This is the portion of the Web that is relevant to users of location-based applications, while they are in an infostation's hoarding area. Such a location-based subspace (or subspace for short) includes Web pages, as well as links connecting them. Obviously, the number of pages in the associated subspace has a strong impact on the content hit ratio. In our experiments, we considered subspaces with 500, 1000, 2000, 3000, and 10000 pages. For example, in the tourist guide

application reported in [7], in total 500 information items were offered. We also varied the average page sizes in order to analyze the page size sensitivity of our scheme. Following the suggestions in [17,22], we considered subspaces with average page sizes of 20 KBytes (pages with text and pictures), 110 KBytes and 230 KBytes (plus 10% resp. 20% of Web pages including video or audio information). Therefore, in combination, we evaluated our hoarding scheme for 15 different subspaces.

For each of these subspaces, we used 5000 logs to build up the information graph and evaluate our scheme. Note that each of these logs models how a single user navigates the corresponding subspace. Based on the experiences reported in [7], we assume an average log length of 100 and vary the length according to a Poisson distribution.

Unfortunately, to our knowledge, a collection of user logs for location-based applications with statistical relevance is not publicly available. In principle, user logs could be extracted from existing proxy logs. However, in order to systematically evaluate our approach, we need to experiment with logs that are associated with differently sized subspaces, differing in number of pages and average page sizes. Therefore, we decided to synthetically generate the 15 subspaces and associated 75,000 log files, using a tool based on the *Web Browsing Model* introduced in [5]. It comprises two sub-models, the *Web Graph Model* and the *Access Behavior Model*. The former models information spaces with the bow-tie-structure found in the Web and document sizes, while the latter models how users typically navigate the Web, taking into account the Web page popularity, the number of requests per session, the periods user visit Web pages, and the way users choose the next Web page. In the literature, we can find substantial research that validates the applied sub-models and distribution functions by means of an extensive analysis of real-world log files. Moreover, we calibrated our model using the log files gathered from our computer center's proxy server in June 2004 and could show that the synthetic access sequences come very close to the real-world logs [5]. Due to the self-similarity in the Web [3, 9], we can apply this model to the Web as a whole, as well as to a smaller fraction, such as a subspace.

From the 5000 logs per subspace, we randomly selected 1000 so-called evaluation logs. For the particular subspace, we ran the experiments for each evaluation log and averaged the results. That is, each plot-point on the curves below is the average of 1000 result values. The remaining 4000 so-called update logs were used to build up the information graph. In order to evaluate how long it takes that our hoarding scheme "learns" a particular access behavior starting from zero knowledge, we varied the number of update logs.

Different average page sizes lead to different user request volumes for the same number of requests. For the sake of a better comparability of our results, we considered the relative hoard cache size, which is defined to be $\frac{\text{hoard cache size}}{\text{user request volume}}$. In our experiments, we used relative hoard cache sizes of $0.1, 0.25, 0.5, 0.75, 1, 2, \ldots 10$.

The parameters of our evaluations are summarized in Table 1.

6.2 Experimental Results

In the following, we will show some characteristic results of our extensive evaluations. First, we will analyze the content hit ratio of our hoarding scheme for various subspaces differing in the number of pages and average page size. Then, we will compare the performance of the proposed approach with that of other hoarding schemes. Finally, we will investigate how fast our approach learns the users' access behavior by varying the number of update logs to build up the information graph. In all figures shown below, the x-axis shows the relative hoard cache sizes, while the content hit ratio is shown on the y-axis.

Table 1: Parameters used for evaluation

Parameter	Value Range
#pages of subspace	500, 1000, 2000, 3000, 10000
Average page sizes [KBytes]	20, 110, 230
Relative hoard cache size	0.1, 0.25, 0.75, 1 ... 10
#update logs	150, 300, 500, 1000, 2000, 3000, 4000
#evaluation logs	1000

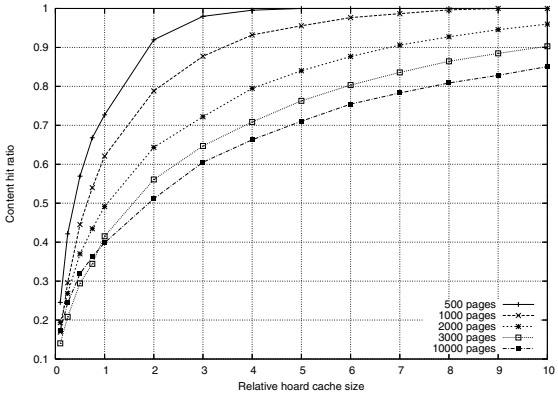


Figure 4: Comparing subspaces with average page size=20 KBytes for varying number of pages

Content Hit Ratio for Different Subspaces. In the following two experiments, we use 4,000 update logs to build up the information graph. In the first one, we analyze the content hit ratios achieved by varying the number of pages in the subspace with a constant average page size of 20 KBytes. The graph depicted in Fig. 4 shows the result curves for subspaces with 500, 1000, 2000, 3000, and 10000 pages. It is not surprising that the content hit ratio decreases with increasing number of subspace pages. However, even in the case of 10,000 pages (from which only 100 are requested on average per user) we achieve a content hit ratio of 80.9% for a hoard cache size that is 8 times the request volume. In other words, we need to hoard only 8% of the subspace volume to achieve a content hit ratio of 80%. In the case of a 1,000 pages subspace, we need to hoard about 2 times the request volume to achieve an 80% content hit ratio. That is, we need to hoard 20% of the subspace volume to achieve this hit ratio. In general, the subspace fraction to be hoarded for reaching a certain hit ratio decreases with increasing subspace size. That is why our scheme shows a significantly better performance for the 10,000 pages subspace than the other schemes, which do not show this characteristic (see below).

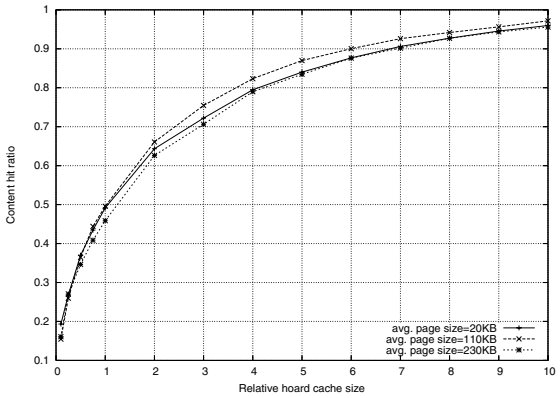


Figure 5: Comparing subspaces with 2000 pages for varying average page sizes

Next, we compare the content hit ratios by varying the average page sizes, leaving the number of subspace pages constant. Fig. 5 depicts the content hit ratios achieved for a 2,000 pages subspace with page sizes of 20, 110, and 230 KBytes. Our results show that the content hit ratio is independent of the average page size. Our experiments with all the other subspaces showed the same characteristic. Consequently, our scheme is stable in terms of varying average page sizes. This nice property is mainly due to the fact that our scheme takes into account page sizes when computing the relevance of a cluster.

Comparison with Other Hoarding Schemes. Now, we compare the hoarding scheme proposed in this paper to three other approaches, which have been briefly described in Section 2: Both, the Least Frequently Used (LFU) scheme decribed in [12], and the Least Recently Used (LRU) scheme, proposed in [13] for hoarding and in [20] for Web caching, do not exploit semantic relationships between data items nor do they allow for clustering. Though the scheme introduced in [6] derives semantic relations based on user access behavior, it does not provide for clustering. We will call it BFS-Scheme since it applies a Breadth First Search to traverse the underlying information graph. Our scheme will be denoted as Clustering-Based Hoarding (CBH) scheme.

In the following, we compare the four schemes for subspaces with 2,000 and 10,000 pages and with an average page size of 20 KBytes. To build up the information graph, we used 4,000 update logs. Fig. 6(a) and Fig. 6(b) show the results of the comparison for 2,000 and 10,000 subspace pages, respectively. For both subspaces, CBH outperforms all the other schemes. For the 2,000 pages subspace, CBH achieves a content hit ratio that is up to factor 1.2 better than LFU, which is the second best. For the 10,000 pages subspace, CBH outperforms LFU, which again is the second best, by a factor of 2.4. We also performed this comparison for the other subspaces, and our results show that the factor, CBH outperforms the other schemes, increases with the subspace size. This is due to the characteristic of CBH pointed out in the discussion of Fig. 4, that the subspace fraction to be hoarded for reaching a certain hit ratio decreases with the subspace size. This characteristic does not hold for LFU, LRU and BFS.

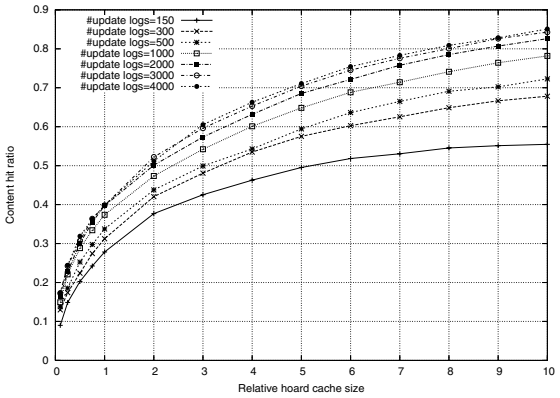


Figure 7: Content hitratio in terms of number of update logs

Impact of Number of Update Logs. In the following, we assume that an infostation starts with zero knowledge, i.e., its information graph is initially empty. We now investigate how fast it learns about the user behavior by using the uploaded logs to update its information graph. For that, we measure the content hit ratio for the varying numbers of update logs specified in Table 1. Fig. 7 shows the results for a 10,000 pages subspace. For 150 and 4,000 update logs, the difference in hit ratio is significant, up to 30%. However, with only 150 update logs, we can already achieve a hit ratio of up to 55.5%. For more than 2,000 update logs, the improvements in hit ratio are only marginal.

7. CONCLUSION

In this paper, we introduced a location-based approach for hoarding semi-structured data based on clustering. We have classified pages into content and transit pages and used this classification to guide our clustering algorithm. In order to maximize the hit ratio, we sorted the clusters according to their relevance per byte rather than per page. We showed by means of experimental evaluation that our hoarding scheme outperforms existing ones, that do not make use of clustering by a factor of more than 2 in terms of content hit ratio. Our results also revealed that the fraction of the subspace to be hoarded for reaching a specific hit ratio, decreases with increasing subspace size. That is why the gain of our scheme compared to others increases with increasing subspace size. Finally, the performance of our scheme is independent of the average page size.

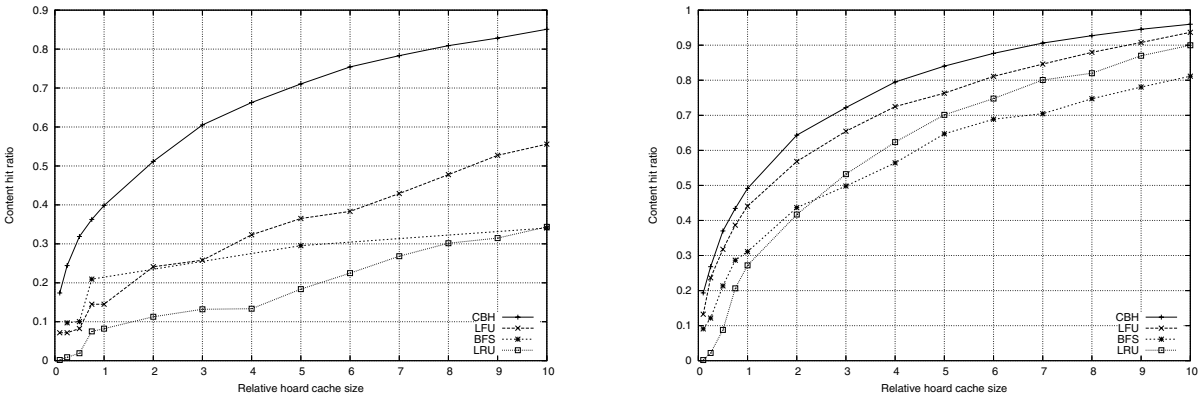
In this paper, we assumed a single interest group (i.e., a single information graph per infostation) when evaluating our scheme. Next, we will investigate the impact of multiple interest groups on the performance.

8. REFERENCES

[1] Martin Arlitt, Diwakar Krishnamurthy, and Jerry Rolia. Characterizing the scalability of a large web-based shopping system. *ACM Transactions on Internet Technology*, 1(1):44–69, 2001.

[2] D. Badrinath, T. Imielinski, R. Frenkiel, and D. Goodman. Nimble: Many-time, many-where communication support for information systems in highly mobile and wireless environments. http://www.cs.rutgers.edu/dataman/nimble/, 1996.

[3] Albert-László Barabási and Eric Bonabeau. Scale-free networks. *Scientific American*, May 2003.



(a) 10,000 subspace pages

(b) 2,000 subspace pages

Figure 6: Comparing clustering and non-clustering schemes

[4] Doug Beeferman and Adam Berger. Agglomerative clustering of a search engine query log. In *KDD '00: Proc. of the 6th ACM SIGKDD Int. Conf. on Knowledge discovery and data mining*, pages 407–416, New York, NY, USA, 2000. ACM Press.

[5] Susanne Bürklen, Pedro José Marrón, Serena Fritsch, and Kurt Rothermel. User centric walk: An integrated approach for modeling the browsing behavior of users on the web. In *Proc. of the 38th IEEE Ann. Simulation Symp. (ANSS'05), San Diego, CA, USA*, 2005.

[6] Susanne Bürklen, Pedro José Marrón, and Kurt Rothermel. An enhanced hoarding approach based on graph analysis. In *Proc. of the IEEE Int. Conf. on Mobile Data Management (MDM 2004), Berkeley, CA, USA*, pages 358–369, 2004.

[7] K. Cheverst, N. Davies, K. Mitchell, A. Friday, and C. Efstratiou. Developing a context-aware electronic tourist guide: Some issues and experiences. In *Proc. of the SIGCHI Conf. on Human factors in Computing systems (CHI 2000)*, 2000.

[8] D. B. Crouch, C. J. Crouch, and G. Andreas. The use of cluster hierarchies in hypertext information retrieval. In *HYPERTEXT '89: Proc. of the 2nd Ann. ACM Conf. on Hypertext*, pages 225–237, New York, NY, USA, 1989. ACM Press.

[9] Stephen Dill, S. Ravi Kumar, Kevin S. McCurley, Sridhar Rajagopalan, D. Sivakumar, and Andrew Tomkins. Self-similarity in the web. In *The VLDB Journal*, pages 69–78, 2001.

[10] Jim Griffioen and Randy Appleton. Reducing file system latency using a predictive approach. In *USENIX Summer*, pages 197–207, 1994.

[11] James J. Kistler and M. Satyanarayanan. Disconnected operation in the coda file system. *ACM Transactions on Computer Systems (TOCS)*, 10(1):3–25, 1992.

[12] Uwe Kubach and Kurt Rothermel. Exploiting location information for infostation-based hoarding. In *Proc. of the 7th Ann. Int. Conf. on Mobile Computing and Networking (MobiCom 2001)*, pages 15–27. ACM Press, 2001.

[13] Geoffrey H. Kuenning, Wilkie Ma, Peter Reiher, and Gerald J. Popek. Simplifying automated hoarding methods. In *Proc. of the 5th ACM Int. workshop on Modeling analysis and simulation of wireless and Mobile systems*, pages 15–21. ACM Press, 2002.

[14] Geoffrey H. Kuenning and Gerald J. Popek. Automated hoarding for mobile computers. In *Proc. of the 16th ACM Symp. on Operating Systems Principles*, pages 264–275. ACM Press, 1997.

[15] Kwong Lai and Zahir Tari and Peter Bertok. Supporting Disconnected Operations Through Cooperative Hoarding. In *Proc. of Int. Conf. on Computer Communications and Networks (ICCCN)*, October 2005.

[16] Xiaoyong Liu and W. Bruce Croft. Cluster-based retrieval using language models. In *SIGIR '04: Proc. of the 27th Ann. Int. ACM SIGIR Conf. on Research and development in information retrieval*, pages 186–193, New York, NY, USA, 2004. ACM Press.

[17] Bruce A. Mah. An empirical model of http network traffic. In *INFOCOM '97: Proc. of the INFOCOM '97. 16th Ann. Joint Conf. of the IEEE Computer and Communications Societies. Driving the Information Revolution*, page 592, Washington, DC, USA, 1997. IEEE Computer Society.

[18] M. J. Mana-Lopez, M. De Buenaga, and J. M. Gomez-Hidalgo. Multidocument summarization: An added value to clustering in interactive retrieval. *ACM Transactions on Information Systems*, 22(2):215–241, 2004.

[19] O. Ratsimor and S. Balaji Kodeswaran and A. Joshi and T. Finin and Y. Yesha. Combining Infrastructure and Ad hoc Collaboration For Data Management in Mobile Wireless Networks. In *Workshop on Ad hoc Communications and Collaboration in Ubiquitous Computing Environments*, 2002.

[20] Stefan Podlipnig and Laszlo Böszörmenyi. A survey of web cache replacement strategies. *ACM Comput. Surv.*, 35(4):374–398, 2003.

[21] Qun Ren and Margaret H. Dunham. Using semantic caching to manage location dependent data in mobile computing. In *Proc. of the 6th Ann. Int. Conf. on Mobile Computing and Networking (MobiCom 2000)*, pages 210–221. ACM Press, 2000.

[22] A. Reyes-Lecuona, E. González-Parada, E. Casilari, and A. Díaz-Estrella. A page-oriented www traffic model for wireless system simulations. In *Proceddings of the 16th Int. Teletraffic Congress (ITC'16), Edinburgh, UK*, pages 1271–1280, 1999.

[23] Carl Tait, Hui Lei, Swarup Acharya, and Henry Chang. Intelligent file hoarding for mobile computers. In *Proc. of the 1st Ann. Int. Conf. on Mobile Computing and Networking (MobiCom 1995)*, pages 119–125. ACM Press, 1995.

[24] N. J. Tuah, M. J. Kumar, and S. Venkatesh. Performance modelling of speculative prefetching for compound requests in low bandwidth networks. In *WOWMOM '00: Proc. of the 3rd ACM int. workshop on Wireless mobile multimedia*, pages 83–92, New York, NY, USA, 2000. ACM Press.

[25] Tao Ye, H.-Arno Jacobsen, and Randy Katz. Mobile awareness in a wide area wireless network of info-stations. In *Proc. of the 4th Ann. ACM/IEEE Int. Conf. on Mobile Computing and Networking (MobiCom 1998)*, pages 109–120. ACM Press, 1998.

[26] Jinsuo Zhang, Abdelsalam (Sumi) Helal, and Joachim Hammer. Ubidata: ubiquitous mobile file service. In *Proc. of the 2003 ACM Symp. on Applied Computing*, pages 893–900. ACM Press, 2003.