Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

SEN

Software Engineering

Software ENgineering

A Study of Integrated Document and Connection Caching in the WWW

Susanne Albers, Rob van Stee

REPORT SEN-E0316 DECEMBER 19, 2003

CWI is the National Research Institute for Mathematics and Computer Science. It is sponsored by the Netherlands Organization for Scientific Research (NWO).
CWI is a founding member of ERCIM, the European Research Consortium for Informatics and Mathematics.

CWI's research has a theme-oriented structure and is grouped into four clusters. Listed below are the names of the clusters and in parentheses their acronyms.

Probability, Networks and Algorithms (PNA)

Software Engineering (SEN)

Modelling, Analysis and Simulation (MAS)

Information Systems (INS)

P.O. Box 94079, 1090 GB Amsterdam (NL)
Kruislaan 413, 1098 SJ Amsterdam (NL)
Telephone +31 20 592 9333
Telefax +31 20 592 4199

ISSN 1386-369X

A Study of Integrated Document and Connection Caching in the WWW

ABSTRACT
Document caching and connection caching are extensively studied problems. In document caching, one has to maintain caches containing documents accessible in a network. In connection caching, one has to maintain a set of open network connections that handle data transfer. Previous work investigated these two problems separately while in practice the problems occur together: In order to load a document, one has to establish a connection between network nodes if the required connection is not already open. In this paper we present the first study that integrates document and connection caching. We first consider a very basic model in which all documents have the same size and the cost of loading a document or establishing a connection is equal to 1. We present deterministic and randomized online algorithms that achieve nearly optimal competitive ratios unless the size of the connection cache is extremely small. We then consider general settings where documents have varying sizes. We investigate a FAULT model in which the loading cost of a document is 1 as well as a BIT model in which the loading cost is equal to the size of the document.

2000 Mathematics Subject Classification: 68W25;68W40
1998 ACM Computing Classification System: F.2.2
Keywords and Phrases: caching;connection caching; online algorithms
Note: Work supported by the Deutsche Forschungsgemeinschaft, Project AL 464/3-1, and by the European Community, Projects APPOL and APPOL II.

A Study of Integrated Document and Connection Caching in the WWW*

Susanne Albers† Rob van Stee‡

Abstract

Document caching and connection caching are extensively studied problems. In document caching, one has to maintain caches containing documents accessible in a network. In connection caching, one has to maintain a set of open network connections that handle data transfer. Previous work investigated these two problems separately while in practice the problems occur together: In order to load a document, one has to establish a connection between network nodes if the required connection is not already open.

In this paper we present the first study that integrates document and connection caching. We first consider a very basic model in which all documents have the same size and the cost of loading a document or establishing a connection is equal to 1. We present deterministic and randomized online algorithms that achieve nearly optimal competitive ratios unless the size of the connection cache is extremely small. We then consider general settings where documents have varying sizes. We investigate a FAULT model in which the loading cost of a document is 1 as well as a BIT model in which the loading cost is equal to the size of the document.

1 Introduction

Recently there has been considerable research interest in document caching [5, 7, 8, 9, 10, 11, 12] and connection caching [2, 3, 4] in networks. In document caching, one has to maintain local caches containing documents available in the network. In connection caching, one has to maintain a set of open network connections that handle data transfer. However, previous work investigated these two problems separately, while in practice they are very closely related.

Consider a computer that is connected to a network. A user working at that computer wishes to access and download documents from other network sites. A downloaded document can be stored in local cache, so that it does not have to be retransmitted when the user wishes to access that document again. Serving requests to documents that are stored locally is much less expensive than transmitting requested documents over the network. Therefore, the local cache, which is of bounded capacity, should be maintained in a careful manner. The transmission of documents in a network is performed using protocols such as TCP (Transmission Control Protocol). If a network node v has to download a document available at node v', then there has to exist an open (TCP) connection between v and v'. If the connection is not already open, it has to be established at a cost. Most networks, such as the Web, today work with persistent connections, i.e. an established connection can be kept open and reused later. However, each network node can only maintain a

*A preliminary version of this paper appeared in Automata, Languages and Programming. 30th International Colloquium (ICALP 2003), LNCS 2719, pp.653–667. Work supported by the Deutsche Forschungsgemeinschaft, Project AL 464/3-1, and by the European Community, Projects APPOL and APPOL II.

†Institut für Informatik, Albert-Ludwigs-Universität, Georges-Köhler-Allee, 79110 Freiburg, Germany. salbers@informatik.uni-freiburg.de.

‡Centre for Mathematics and Computer Science (CWI), Kruislaan 413, NL-1098 SJ Amsterdam, The Netherlands. Rob.van.Stee@cwi.nl.

limited number of open connections and the collection of open connections can be viewed as a *connection cache*. The goal is to maintain this cache so that the connection establishment cost is as small as possible.

Clearly, caching decisions made on the document and connection levels heavily affect each other. Evicting a document d from the document cache at node v has a very negative effect if the connection between node v and node v', where d is originally stored, is already closed. When d is requested again, one has to pay the connection establishment cost in addition to the necessary document transmission cost. A similar overhead occurs if a connection is closed that is frequently needed for data transfers. Therefore document and connection caching algorithms should coordinate their decisions. This can considerably improve the system's performance, i.e. the user perceived latency as well as the network congestion are reduced.

In this paper we present the first study of integrated document and connection caching. Formally, we consider a network node v. The node has two caches: one for the documents, also called *pages*, and one for the open connections currently maintained to other nodes. A sequence of *requests* must be served. Each request specifies a document d that the user at our network node wishes to access. If d resides in the document cache, then the request can be served at 0 cost. Otherwise a fault occurs and the request must be served by downloading d into the document cache at a cost of $cost(d) > 0$. Suppose that d is originally stored at network node v'. To load d into the document cache, an open connection must exist between v and v'. If the connection is already open, no cost is incurred. Otherwise the connection has to be established at a cost of $cost(v, v')$. The goal is to serve the request sequence so that the total cost is as small as possible.

The integrated document and connection caching problem is inherently online in that each request must be served without knowledge of future requests. We use competitive analysis to analyze the performance of online algorithms. We denote the cost of an algorithm $\mathcal{A}$ on a request sequence σ by $\mathcal{A}(\sigma)$. The optimal cost to serve this sequence is denoted by $\text{OPT}(\sigma)$. The goal of an online algorithm $\mathcal{A}$ is to minimize the competitive ratio $\mathcal{R}(A)$, which is defined as the smallest value $\mathcal{R}$ that satisfies $\mathcal{A}(\sigma) \leq \mathcal{R} \cdot \text{OPT}(\sigma) + a$, for any request sequence σ and some constant a independent of σ.

We remark here that a problem similar to that defined above arises in distributed databases. There, a user may have a file/page cache as well as a cache with pointers to files allowing fast access.

Previous work: As mentioned above document and connection caching have separately been the subjects of extensive research. There is a considerable body of work on document caching problems, see e.g [5, 7, 8, 9, 10, 11, 12]. However, the papers ignore that in a network setting, one may have to open a connection to load a document. If all documents have the same size and a loading cost of 1, which is the classical paging problem, the best competitive ratio of deterministic online algorithms is equal to k, where k is the number of documents that can be stored simultaneously in cache [11]. This competitiveness is achieved by the popular LRU (Least Recently Used) and FIFO (First-In First-Out) replacement strategies. On a fault, LRU evicts the page that was requested least recently and FIFO evicts the page that has been in cache longest. Fiat et al. [7] presented an elegant randomized paging algorithm called MARK that is $2H_k$-competitive against oblivious adversaries, where H_k is the k-th Harmonic number. More complicated algorithms that achieve an optimal competitiveness of H_k were given in [1, 10]. Irani [9] initiated the algorithmic study of the document caching problem when documents have different sizes. She considered a FAULT model where the loading cost of each document is equal to 1 as well as a BIT model, where the loading cost is equal to the size of the document. She presented randomized $O(\log^2 k)$-competitive online algorithms for both settings. Young [12] gave a deterministic k-competitive online algorithm for a general cost model where the loading cost is an arbitrary non-negative value. Recently Feder et al. [5] studied a document caching problem where requests can be reordered. They concentrate on the case that the cache can hold one document. Gopalan et al. [8] study document caching in the Web when documents have expiration times. They assume all documents have the same size and a loading cost of 1.

Cohen et al. [3, 4] introduced the connection caching problem. The input of the problem is a sequence

of requests for TCP connections that must be established if not already open. Cohen et al. considered a distributed setting where requests occur at different network nodes. They gave deterministic k-competitive and randomized $O(H_k)$-competitive online algorithms if all connections incur the same establishment cost. Here k is the maximum number of connections that a network node can keep open simultaneously. The case that connections can have varying establishment costs was considered in [2].

Our contribution: We investigate document and connection caching in an integrated manner. In the following let k be the number of documents that can be stored in the document cache and k' be the number of connections that can be kept open. We start by studying a basic setting in which all documents have the same size and a loading cost of 1; the connections have an establishment cost of 1. We present a deterministic online algorithm that achieves a competitive ratio of $k+4$ if $k' \geq k$ and a ratio of $\min\{2k-k'+4, 2k\}$ if $k' < k$. Our algorithm uses LRU for the document cache and a phase based replacement strategy that tries to keep connections of documents that may be evicted soon. We develop a lower bound on the performance of any deterministic online algorithm which implies that our algorithm is nearly optimal if k' is not extremely small. We also consider randomized online algorithms and prove that by replacing LRU by a randomized Marking strategy we obtain a competitive ratio of $2H_k + \min\{2H_k, 2(k-k')+4\}$.

Additionally we investigate the problem that pages have varying sizes. If all documents have a loading cost of 1, which corresponds to Irani's FAULT model, we achieve a competitive ratio of $(4k+14)/3$ if $k' \geq k$ and of $2k - 2k'/3 + 14/3$ if $k' < k$. Finally we consider a BIT model where the loading cost of a document is equal to the size of the document and the connection establishment cost is c, for some constant c. Here we prove a competitiveness of $(k+5)(c'+1)/2$ if $k' \geq k$, where $c' = c/s$ and s is the size of the smallest document ever requested. If $k' < k$, the competitiveness is $(2k-k'+5)(c'+1)/2$.

Finally we consider a distributed scenario, where requests can occur at different network nodes. We show that no deterministic online algorithm can in general be better than $2k$-competitive, where k is the maximum number of documents that can be stored at any network node. A competitive ratio of $2k$ is easily achieved by an online algorithm that uses a k-competitive paging algorithm for the document cache and any replacement strategy for the connection cache.

2 Algorithms for the basic model

In this section we study a very basic scenario where all documents have the same size. Loading a missing document costs 1 and establishing a connection also costs 1.

2.1 Deterministic algorithms

We present a deterministic online algorithm ALG for our basic setting. ALG works in phases. Each phase is defined as a maximal subsequence of requests to k distinct pages, which starts after the previous phase finishes (the first phase starts with the first request). Within each phase ALG works as follows.

At the beginning of each phase, evict all connections that were not used in the previous phase. On a page fault, use LRU to determine which page to evict from the page cache. On a connection fault, if there is a free slot in the cache, use it; otherwise, use MRU (Most Recently Used) to determine which connection to evict.

For ease of exposition, we first consider the case where the size of the connection cache is at least the same size as the page cache, i.e. $k' \geq k$. We then extend our analysis to the case $k' < k$.

Theorem 1 *If $k' \geq k$, then $\mathcal{R}(\text{ALG}) \leq k+4$.*

Proof. Consider a request sequence σ. We first study the case that $k' = k$. Suppose there are $N + 1$ phases, numbered $0, 1, \ldots, N$. For phase i, denote the number of page requests that cause a page fault by f_i; the number of page requests that do not cause a page fault by p_i (these pages were requested in the previous phase by definition of LRU); the number of MRU faults m_i, and the number of holes *created* by h_i (i. e. the number of connections evicted at the start of phase i). Define $F = \sum_{i=1}^{N} f_i$, $M = \sum_{i=1}^{N} m_i$, $H = \sum_{i=2}^{N} h_i$ and $P = \sum_{i=1}^{N} p_i$. (We ignore phase 0.) Note $h_1 = 0$ and $f_i + p_i = k$ for each phase i.

Each hole that is created, is filled at most once, and this happens on a connection fault. (It is possible that some holes are never filled.) Thus the number of connection faults that cause holes to be filled is at most H. Furthermore, the remaining connection faults are exactly the connection faults where MRU is applied; this happens M times. Thus

$$\text{ALG}(\sigma) \leq F + M + H = kN + M + H - P. \tag{1}$$

Note that our algorithm is defined in such a way that the number of page faults is independent of the number of connection faults or the decisions as to which connections are evicted. The page cache is simply maintained by LRU. By definition of LRU, there must be one OPT page fault in each phase. Thus

$$\text{OPT}(\sigma) \geq N. \tag{2}$$

Each phase can be visualized as follows. The connection cache is at all times divided into two sets, PREVIOUS and CURRENT. Here PREVIOUS contains the connection slots that were not (yet) used in this phase, while CURRENT contains the connection slots that were used in the current phase. At the start of each phase, CURRENT is empty and PREVIOUS contains all k slots. Note that some of these slots may contain holes, in case a connection was evicted that was not used in the previous phase.

For each page fault in a phase, there are two possibilities:

1. No connection fault:
   (a) A not yet used connection slot is used for the first time in this phase (this connection was also used in the previous phase);
   (b) A connection slot already used in the current phase is used again (two or more pages are at the same node).
2. Connection fault occurs:
   (a) A hole is filled: a not yet used connection slot is used for the first time in this phase;
   (b) A connection slot already used in the current phase is used again by MRU;
   (c) (special case) A connection slot not yet used in the current phase is used by MRU.

Case 2.(c) can only occur if the very first page fault in a phase causes a connection fault; for a later page fault that also causes a connection fault, MRU always uses a slot that was already used in the current phase. From this list we have that only in cases 1.(a), 2.(a) and 2.(c), a connection slot moves from the set PREVIOUS to the set CURRENT.

Consider a phase $i > 0$. Suppose Case 2.(c) does not occur, and there are $m_i > 0$ MRU faults in phase i. Then at least m_i times, a connection slot already in CURRENT is used again. Hence at most $f_i - m_i$ times a connection slot moves from PREVIOUS to CURRENT. Therefore, at the end of phase i, there are at least $k - f_i + m_i$ connection slots still in PREVIOUS.

The pages requested in phase i can be divided into four groups:

1. pages that did not cause a page fault (p_i);

2. pages that caused a page fault, but no connection fault;

3. pages that caused a hole in the connection cache to be filled;

4. pages that caused a connection slot to be used again by MRU (m_i).

Every connection slot that at some point in phase i contains a connection to a page in group 2 or 3 (note that this may change later in the phase due to the use of MRU), is in CURRENT at the end of the phase. The other connection slots contain connections to pages that were either not requested in phase i (but were requested in phase $i-1$, or they would have been evicted before), or that did not cause a fault. This last possibility occurs p_i times, so there are at least $k - f_i + m_i - p_i = m_i$ pages that are not requested again. This implies there are at least $k + m_i$ distinct pages requested in phase i and phase $i-1$. Therefore OPT has at least m_i faults in phases $i-1$ and i.

If Case 2.(c) does occur, then there were no holes at the start of phase i. Then the connections to the pages requested in phase $i-1$ must all be distinct, $m_{i-1} = 0$ and $h_i = 0$. At the start of phase i, a connection slot moves from PREVIOUS to CURRENT using MRU. Case 2.(c) does not occur in the rest of the phase. Thus at the end of phase i, we have that there are at least $k - f_i + m_i - 1$ connection slots still in PREVIOUS. These slots correspond to connections that were used in the previous phase but not in this one, implying $k - f_i + m_i - p_i - 1 = m_i - 1$ pages that were requested in phase $i-1$ but not in i. Then OPT has at least $m_i - 1$ faults in phases $i-1$ and i. Moreover, it has at least one fault in phases $i-2$ and $i-1$, and $1 = m_{i-1} + 1$. By amortizing the cost, we find that OPT has at least m_i faults for every pair of phases $i-1$ and i.

Thus $\text{OPT}(\sigma) \geq \sum_{i \text{ odd}} m_i$, and $\text{OPT}(\sigma) \geq \sum_{i \text{ even}} m_i$. This implies that

$$\text{OPT}(\sigma) \geq \frac{1}{2} \sum_{i>0} m_i = \frac{M}{2}. \tag{3}$$

The connections still in PREVIOUS at the end of phase i are evicted and become h_{i+1} holes. At most p_i of them lead to pages that were requested without a fault. Thus there are at least $k + h_{i+1} - p_i$ distinct pages requested in phases i and $i-1$. This gives another bound for the cost of OPT:

$$\text{OPT}(\sigma) \geq \frac{1}{2} \sum_{i>0} (h_{i+1} - p_i) \geq \frac{H - P}{2} \tag{4}$$

Combining (1), (2), (3) and (4) gives

$$\text{ALG}(\sigma) \leq kN + M + H - P \leq k \cdot \text{OPT}(\sigma) + 2\text{OPT}(\sigma) + 2\text{OPT}(\sigma) = (k+4)\text{OPT}(\sigma).$$

This proves the ratio. It can be seen that the proof also holds for $k' > k$. □

Theorem 2 *If $k' < k$, then $\mathcal{R}(\text{ALG}) \leq \min(k + 4 + (k - k'), 2k)$.*

Proof. Clearly, $\mathcal{R}(\text{ALG}) \leq 2k$ since ALG has at most $2k$ faults per phase (k connection faults and k page faults). We still have (2) and (4) by the exact same reasoning as in the proof of Theorem 1.

For m_i, we have again that each time that MRU is applied, no connection moves from PREVIOUS to CURRENT (unless Case 2.(c) occurs). So at most $f_i - m_i$ times a connection moves from PREVIOUS to CURRENT. Therefore, at the end of the phase, at least $k' - f_i + m_i$ connections are still in PREVIOUS. At most p_i of them refer to pages requested without a fault in phase i, so at least $k' - f_i + m_i - p_i = k' - k + m_i$ pages are requested in phase $i-1$ but not in phase i. Therefore there are at least $m_i + k'$ distinct pages requested in these two phases, and OPT has at least $m_i - (k - k')$ faults.

If Case 2.(c) occurs, there are only at least $k' - (f_i - (m_i - 1)) = m_i - (k - k') - 1$ connections still in PREVIOUS at the end. However, in that case we have $m_{i-1} \leq k - k'$ since there were no holes. Therefore $m_{i-1} - (k - k') \leq 0$ and we can amortize as before.

We therefore find

$$\text{OPT}(\sigma) \geq \frac{M - (k - k')N}{2}. \tag{5}$$

Using (2), this implies $M \leq 2\text{OPT}(\sigma) + (k - k')N \leq (k - k' + 2)\text{OPT}(\sigma)$. Therefore in this case

$$\text{ALG}(\sigma) \leq ((k + 2) + (k - k' + 2))\text{OPT}(\sigma) \leq (2k - k' + 4)\text{OPT}(\sigma).$$

This proves the lemma. □

2.2 Randomized algorithms

For the standard paging problem, the randomized algorithm MARK is $2H_k$-competitive, where H_k is the k-th Harmonic number [7]. Moreover, no randomized algorithm can have a competitive ratio less than H_k. The MARK algorithm processes a request sequence in phases. At the beginning of each phase, all pages in the memory system are unmarked. Whenever a page is requested, it is *marked*. On a fault, a page is chosen uniformly at random from among the unmarked pages in cache, and that page is evicted. A phase ends when all pages in cache are marked and a page fault occurs. Then, all marks are erased and a new phase is started.

In our algorithm ALG we substitute MARK for LRU to get a randomized algorithm. However, in this case it is also necessary to evict connections less greedily to get a good performance. In particular, at the start of a phase we will not evict any connections that are associated with pages requested in the previous phase. Note that some of these connections may not have been used in that phase, because the relevant page might not have caused a page fault.

Theorem 3 *For the randomized version of* ALG *and $k' \geq k$, we have $\mathcal{R}(\text{ALG}) \leq 2\mathrm{H}_k + 4$. For $k' < k$, we have. $\mathcal{R}(\text{ALG}) \leq 2\mathrm{H}_k + \min(2\mathrm{H}_k, 4 + 2(k - k'))$.*

Proof. We analyze this algorithm very similarly to the original analysis of MARK [7] and to the analysis in Section 2.1. We define q_i as the number of *new* pages requested in phase i. A page is new if it is not in the cache at the start of the phase. We define h_i, m_i, H and M as before and write $Q = \sum q_i$. Then by [7],

$$\text{ALG}(\sigma) \leq \mathrm{H}_k Q + H + M.$$

Moreover, $\text{OPT}(\sigma) \geq Q/2$.

Following the proof of the deterministic case, we now have that every connection slot that at some point in phase i contains a connection to a page in group 1, 2 or 3 (note that this may change later in the phase due to the use of MRU), is in CURRENT at the end of the phase. Therefore any connections that are still in PREVIOUS at that time (which get evicted and form holes) must be to pages not requested in the phase. Therefore $\text{OPT}(\sigma) \geq H/2$.

Suppose $k' \geq k$. Due to the randomization, we do not know whether or not Case 2.(c) occurs in a phase. However, as observed in the proof of the deterministic algorithm, we can amortize the offline faults if 2.(c) occurs to get the bound $\text{OPT}(\sigma) \geq M/2$. Therefore analogously to in the proof of Theorem 1, we have

$$\mathcal{R}(\text{ALG}) \leq 2\mathrm{H}_k + 4.$$

We now consider the case $k' < k$. The only change is that the bound $\text{OPT}(\sigma) \geq M/2$ is replaced by

$$\text{OPT}(\sigma) \geq \frac{M - (k - k')N}{2} \geq \frac{M - (k - k')Q}{2},$$

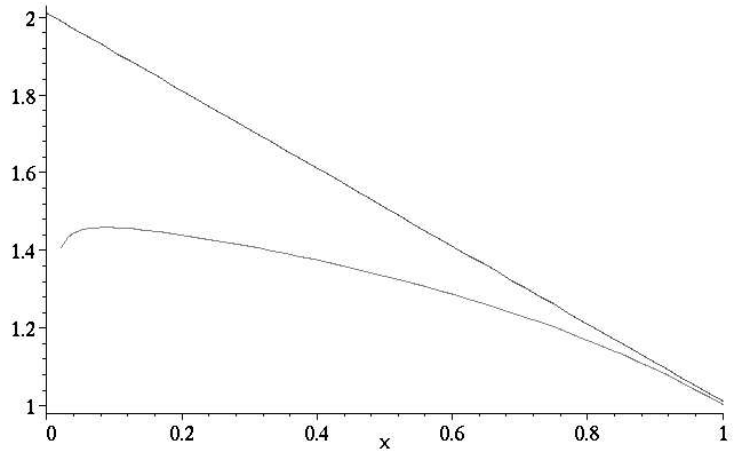

Figure 1: The upper and lower bound: x-axis is k'/k, y-axis is $\mathcal{R}/k$

where we have used $Q \geq N$, which follows from the fact that there must be at least one new page in every new phase by definition of the phases. This gives us

$$\mathcal{R}(\text{ALG}) \leq \frac{\mathrm{H}_k Q + H + M}{\text{OPT}(\sigma)} \leq 2\mathrm{H}_k + 4 + 2(k - k').$$

However, since the number of connection faults, $H + M$, is also upper bounded by the number of page faults $\mathrm{H}_k Q$, we find

$$\mathcal{R}(\text{ALG}) \leq 2\mathrm{H}_k + \min(2\mathrm{H}_k, 4 + 2(k - k')).$$

□

3 Lower bounds

We present a lower bound on the performance of any deterministic online algorithm. The lower bound of Theorem 4 implies that if k' is not too small, our deterministic algorithm given in the last section is nearly optimal. Figure 1 depicts the lower as well as the upper bound.

Theorem 4 *Suppose $k' \geq 2$ and let $\alpha = k'/k$. Then for any online algorithm $\mathcal{A}$, we have*

$$\mathcal{R}(\mathcal{A}) \geq (k+1)\left(\frac{\alpha k - 1}{\alpha k} + \frac{1-\alpha}{2 - \alpha + 3/k}\right).$$

Proof. We construct a lower bound as follows. We make use of $k+1$ pages that are stored at $k+1$ distinct nodes. Consider an online algorithm $\mathcal{A}$. Each page request in the sequence is to the (unique) page that $\mathcal{A}$ does not have in its cache. The sequence is divided into phases. In each phase, we count the number of distinct pages that have been requested in that phase; the first request to the $k+1$st distinct page is defined to be the start of the next phase. Since the connection cache has size k', $\mathcal{A}$ must have at least $k - k'$ connection faults in each phase. We define $\alpha = k'/k$, so that $k' = \alpha k$. We will write the average length of a phase as pk, where $p \geq 1$. The offline algorithm uses one of the following strategies depending on p.

Strategy 1. (For large p.) The first strategy is to always use LFD for the requested pages. We then count the number of offline page faults for each of the $k+1$ pages, and put $k'-1$ connections to pages on which the most offline faults occur, in the connection cache. This part of the connection cache is fixed during the entire processing of the request sequence. The last slot is used for connection faults on the remaining $k+1-(k'-1) = k-k'+2$ pages.

Consider $k+1$ phases. There are at most $k+1$ offline faults, and on average at most $k-k'+2$ of them are on pages of which the connections are not in the connection cache at all times. Thus there are on average at most $2k-k'+3$ offline faults on $k+1$ phases.

Strategy 2. (For small p.) The second strategy begins by counting the number of requests to each page over the entire request sequence. Then, the $k-k'+1$ pages that are requested the most often, are put in the page cache at the beginning, and the k' connections to the remaining pages are put in the connection cache. The entire connection cache is fixed throughout the sequence. The offline algorithm now uses LFD on the k' pages for which the connections are in the connection cache, and only uses the $k'-1$ slots in the page cache that do not contain the $k-k'+1$ most often requested pages. It has no connection faults at all.

Consider $(k+1)(k'-1)$ phases. These contain on average $(k+1)(k'-1)pk$ requests by definition of p. Thus, each page is requested on average $(k'-1)pk$ times. The k' pages that are requested the least overall, must then be requested at most $k'(k'-1)pk$ times on average at most. Since the offline algorithm has at most one fault every $k'-1$ requests to this subset of pages, there are $k'pk$ offline faults.

Solving for p, we find that these two strategies have the same number of faults if

$$p = \frac{\alpha k - 1}{\alpha k}\left(2 - \alpha + \frac{3}{k}\right). \tag{6}$$

As long as this value is at least 1, we can use the first offline strategy if p is greater than the threshold, and the second strategy otherwise. The number of on-line faults in one phase must be at least $pk + (k-k')$ on average. This implies a competitive ratio of at least

$$\frac{(pk+k-k')(k+1)(\alpha k-1)}{k'pk} = (k+1)\left(\frac{\alpha k-1}{\alpha k} + \frac{1-\alpha}{2-\alpha+3/k}\right).$$

Note that the threshold in (6) is greater than 1 for $k \geq k' \geq 2$. □

We can show that the analysis of our algorithm ALG is asymptotically tight for $k' = 1$. Note that ALG behaves exactly like LRU in this case. This implies that even for $k' = 1$ it is nontrivial to find an algorithm with competitive ratio close to k.

Lemma 1 *For $k' = 1$, we have $\mathcal{R}(\text{ALG}) \geq 2k-2$.*

Proof. We use a set of pages numbered $1, 2, \ldots, k+1$ and request them cyclically. All the odd pages are at some node v_1 while the even pages are at another node v_2. It can be seen that our algorithm has a connection fault on every request, thus it has $2k$ faults per phase.

We now describe an off-line algorithm to serve this sequence. This algorithm only faults on pages in v_1, and each time evicts the page from that node that will be requested the furthest in the future. All pages in v_2 are in the cache at all times. Suppose k is even, then there are $k/2$ slots available in the cache for $k/2+1$ pages. Thus this off-line algorithm has a fault once every $k/2$ requests to pages in v_1.

Consider $k+1$ phases. It contains $k(k+1)$ requests, exactly k per page. Thus there are $2(k/2+1) = k+2$ offline faults in total, giving a competitive ratio of

$$\frac{2k(k+1)}{k+2} = 2k - \frac{2k}{k+2} \geq 2k-2.$$

For odd k, there is one off-line fault per $(k-1)/2$ requests to pages in v_1. In $k-1$ phases there are $k(k-1)$ requests, thus $k(k-1)/2$ requests to pages in v_1 and in total k offline faults. This gives a ratio of exactly $2k-2$. □

4 Generalized models

In this section we study generalized problem settings in which the documents can have different sizes. For the standard multi-sized paging problem, the algorithm LRU is $(k+1)$-competitive in both the BIT and the FAULT model [6]. Here k is defined as the maximum number of pages that can fit in the cache, i.e. $k = K/s$ where K is the size of the cache (in bits) and s is the size of the smallest possible page. It is nontrivial to extend the analysis of our algorithm to these models.

In both models, a *phase* is now defined as a maximal subsequence of requests to a minimal volume of distinct pages that is larger than K. Thus there are at most $k+1$ page faults in a phase.

4.1 The Fault Model

For the FAULT model, we need to consider the number of pages requested in each phase, which can be less than k.

Theorem 5 *In the* FAULT *model,* $\mathcal{R}(\text{ALG}) \leq (4k+14)/3$ *for* $k' \geq k$ *and* $\mathcal{R}(\text{ALG}) \leq 2k - \frac{2}{3}k' + \frac{14}{3}$ *for* $k' < k$.

Proof. Suppose $k' = k$. Denote the number of pages requested in phase i by Φ_i. Write $\Delta_i = \Phi_i - \Phi_{i-1}$.

If there are m_i connection faults where MRU is applied, then m_i times a connection slot remains in CURRENT. Thus at most $k+1-m_i$ times a connection slot moves from PREVIOUS to CURRENT, and at least $m_i - 1$ connection slots are still in PREVIOUS at the end of the phase. These connections lead to at least $m_i - 1$ pages that were requested in phase $i-1$ but not in phase i.

Denote the set of pages requested in phase $i-1$ but not in phase i by F. Denote the set of pages requested in phase i by S. We partition F in two sets: F_1 contains the pages that OPT faults on, F_2 contains the rest. Consider the set F_2. OPT does not fault on these pages and thus has them in its cache at the start of phase $i-1$. This means that some pages in S are not yet in its cache and need to be loaded later.

Write the number of OPT faults in these two phases as $m_i - 1 - x$. If $x \leq 0$, we are done. Otherwise, F_2 contains $z \geq x > 0$ pages. OPT has exactly $z - x$ faults on the set S, that is, $z - x$ pages are loaded to come "in the place of" the z pages in F_2 (OPT does not necessarily replace exactly these pages in the cache). Since at most $k+1$ pages were requested in phase $i-1$, the set S then contains at most $k+1-x$ pages, i.e. our algorithm has at most $k+1-x$ page faults in phase $i+1$.

That is, if OPT has x faults less than $m_i - 1$ in phases $i-1$ and i, then our algorithm has (at least) x faults less than $k+1$ in phase i. Writing the number of OPT faults as $m_i - 1 - x_i$ in all cases where it is less than $m_i - 1$, this gives

$$\text{OPT}(\sigma) \geq \frac{M - N - X}{2},$$

where $X = \sum x_i$. (That is, all values x_i are positive.)

We can treat the holes that are created in the same way to find

$$\text{OPT}(\sigma) \geq \frac{H - P - X}{2}.$$

Finally we also still have $\text{OPT} \geq N$. We have

$$\text{ALG}(\sigma) \leq (k+1)N - X + M + H - P \quad \text{and} \quad \text{ALG} \leq 2((k+1)N - X),$$

where the second inequality follows since ALG has at most one connection fault for each page fault.

Thus if $X \geq \frac{kN-4}{3}$, we find that the competitive ratio is at most $4k/3 + 14/3$. On the other hand, if $X < \frac{kN-4}{3}$, then

$$\text{ALG}(\sigma) \leq (k+1)\text{OPT}(\sigma) + 4\text{OPT}(\sigma) + X \leq (k + 5 + k/3 - 4/3)\text{OPT}(\sigma) = \frac{4k+14}{3}\text{OPT}(\sigma).$$

This analysis can easily be extended to the case $k' < k$ as before.

Similarly to in Theorem 2, we find that in this case there are at least $k' - k + m_i - 1$ pages requested in phase $i-1$ but not in phase i. Writing the number of OPT faults as $k' - k + m_i - 1 - x$ and distinguishing cases, we now find

$$\text{OPT}(\sigma) \geq \frac{M - (k - k' + 1)N - X}{2}.$$

The other bounds do not change, giving $\mathcal{R}(\text{ALG}) \leq 2k - \frac{2}{3}k' + \frac{14}{3}$. □

4.2 The BIT model

In this section we investigate a BIT model in which the cost of loading a document is equal to the size of the document. We also assume that the cost of establishing a connection is equal to c, for some constant $c > 0$.

Theorem 6 *In the* BIT *model,* $\mathcal{R}(\text{ALG}) \leq \frac{k+5}{2}(c'+1)$ *for* $k' \geq k$*, where* $c' = c/s$ *is the cost of a connection fault divided by the size of the smallest possible page. For* $k' < k$, $\mathcal{R}(\text{ALG}) \leq \frac{2k+5-k'}{2}(c'+1)$.

Proof. Denote the average phase length by $K + \delta$ for some $\delta > 0$. Denote the average number of MRU faults in a phase by m and the average number of bits worth of old pages that are implied by m', then $m' \geq ms$. Denote the average number of pages on which there is no fault in a phase by p and the average number of bits that are requested without fault by p', then $p' \geq ps$. Finally, denote the average number of holes created in a phase by h. Denote the cost of a single connection fault by c and write $c' = c/s$. Similarly to in the previous section, it can be seen that for the average cost in a phase we have $\text{ALG}/s \leq k + \delta/s + (m+h)c/s - p'/s$ and $\text{OPT}/s \geq \max(\max(1, \delta/s), m/2, h - p/2)$. Here the first maximum in the second equation follows from $\sum_i \max(\delta_i, s)/Ns \geq \max(Ns, N\delta)/Ns = \max(1, \delta/s)$, where $K + \delta_i$ is the amount of bits from distinct requests requested in phase i.

Since the number of connection faults in a phase is bounded from above by the number of page faults, we have

$$m + h \leq \frac{K + \delta - p'}{s} \Rightarrow h \leq k + \frac{\delta}{s} - p - m \leq (k+1)\frac{\text{OPT}}{s} - p - m. \tag{7}$$

We also have $h \leq 2\frac{\text{OPT}}{s} + p$. Note that $2\frac{\text{OPT}}{s} + p = (k+1)\frac{\text{OPT}}{s} - p - m \Rightarrow 2p = (k-1)\text{OPT}/s - m$. Suppose $p \leq ((k-1)\text{OPT}/s - m)/2$. (The other case is handled similarly.) Then

$$\begin{aligned}
\frac{\text{ALG}}{s} &\leq (k+1)\frac{\text{OPT}}{s} + mc' + hc' - p \leq (k+1)\frac{\text{OPT}}{s} + mc' + 2\frac{\text{OPT}}{s}c' + p(c'-1) \\
&\leq (k+1)\frac{\text{OPT}}{s} + mc' + 2\frac{\text{OPT}}{s}c' + (c'-1)((k-1)\frac{\text{OPT}}{s} - m)/2 \\
&\leq (k+1)\frac{\text{OPT}}{s} + (c'+1)m/2 + 2\frac{\text{OPT}}{s}c' + (c'-1)(k-1)\frac{\text{OPT}}{2s} \\
&\leq (k + c' + 2 + 2c' + \frac{(c'-1)(k-1)}{2})\frac{\text{OPT}}{s} = \frac{(k+5)(c'+1)}{2} \cdot \frac{\text{OPT}}{s}.
\end{aligned}$$

For $k' < k$, similarly to all previous cases we now find $\text{OPT}(\sigma)/s \geq (m - (k - k'))/2$ and $\mathcal{R}(\text{ALG}) \leq \frac{(2k-k'+5)(c'+1)}{2}$. □

Hence the competitive ratio grows linearly with k and with c (c'). The reason for this is that we cannot identify connection faults by OPT; it is conceivable that OPT never has a connection fault.

5 The distributed setting

We finally study the distributed problem setting where requests can occur at various network nodes. Again, each node has a document cache and a connection cache. Here, a request is specified by a pair (v, d), indicating that document d is requested by the user at node v. The cost of serving requests is the same as before. The crucial difference is in the usage of connections. An open connection between nodes v and v' can be used for downloading documents from v to v' as well as from v' to v. However, if one of the nodes of the connection decides to close the connection, then the connection cannot be used by the other node either. Hence, the connection cache configurations affect each other.

Theorem 7 *In the distributed problem setting, no deterministic online algorithm can achieve a competitive ratio smaller than $2k/(1 + 1/k')$, where k is the size of the largest document cache and k' is the maximum number of connections that a network node can keep open.*

Proof. Consider a node v at which $k + 1$ documents are stored. Additionally we have $k' + 1$ nodes $v_i, 1 \leq i \leq k' + 1$, Each node in the network has a document cache of size k and a connection cache of size k'. Requests are generated as follows. At any time one of the connections (v, v_i) is closed in the configuration of an online algorithm A because v kan only maintain k' open connections and a connection is open only if it is cached by both of its endpoints. An adversary generates a request at this node v_i for the document that is currently not stored in A's document cache at v_i. Suppose that a request sequence consists of m requests and that m_i requests were generated at $v_i, 1 \leq i \leq k' + 1$. The online cost is equal to $2m$. An optimal offline algorithm has at most $\lceil \frac{m_i}{k} \rceil$ document faults at v_i and hence no more than $\frac{m}{k} + k' + 1$ document faults in total. Furthermore an optimal algorithm can maintain the connection cache at v in such a way that at most $\lceil (\frac{m}{k} + k' + 1)/k' \rceil$ connection faults occur. Thus as $m \to \infty$, the ratio of the online to offline cost tends to $2/(\frac{1}{k} + \frac{1}{kk'}) = 2k(1 + 1/k')$. □

Note that a competitive ratio of $2k$ is achieved by any caching algorithm that uses a k-competitive paging strategy for the document cache and any replacement rule for the connection cache.

6 Conclusions

In this paper we studied integrated document and connection caching in a variety of problem settings. An open question left by our work is to find a better algorithm for the case where the connection cache is very small (relative to k). We conjecture that the true competitive ratio for this problem should be close to k.

References

[1] D. Achlioptas, M. Chrobak, and J. Noga. Competitive analysis of randomized paging algorithms. *Theoretical Computer Science*, 234:203–218, 2000.

[2] S. Albers. Generalized connection caching. In *Proceedings of the Twelfth ACM Symposium on Parallel Algorithms and Architectures*, pages 70–78. ACM, 2000.

[3] E. Cohen, H. Kaplan, and U. Zwick. Connection caching. In *Proceedings of the 31st ACM Symposium on the Theory of Computing*, pages 612–621. ACM, 1999.

[4] E. Cohen, H. Kaplan, and U. Zwick. Connection caching under various models of communication. In *Proceedings of the Twelfth ACM Symposium on Parallel Algorithms and Architectures*, pages 54–63. ACM, 2000.

[5] T. Feder, R. Motwani, R. Panigraphy, and A. Zhu. Web caching with request reordering. In *Proceedings 13th ACM-SIAM Symposium on Discrete Algorithms*, pages 104–105, 2002.

[6] A. Feldman, R. Karp, M. Luby, and L. A. McGeoch. Personal communication cited in [9].

[7] A. Fiat, R.M. Karp, M. Luby, L.A. McGeoch, D.D. Sleator, and N.E. Young. Competitive paging algorithms. *Journal of Algorithms*, 12(4):685–699, Dec 1991.

[8] P. Gopalan, H. Karloff, A. Mehta, M. Mihail, and N. Vishnoi. Caching with expiration times. In *Proceedings 13th ACM-SIAM Symposium on Discrete Algorithms*, pages 540–547, 2002.

[9] S. Irani. Page replacement with multi-size pages and applications to web caching. In *Proceedings 29th ACM Symposium on Theory of Computing*, pages 701–710, 1997.

[10] L. McGeoch and D. Sleator. A strongly competitive randomized paging algorithm. *J. Algorithms*, 6:816–825, 1991.

[11] D. Sleator and R. E. Tarjan. Amortized effi ciency of list update and paging rules. *Communications of the ACM*, 28:202–208, 1985.

[12] N. Young. On-line fi le caching. In *Proceedings 9th ACM-SIAM Symposium on Discrete Algorithms*, pages 82–86, 1998.