An Evaluation of Document Prefetching in a Distributed Digital Library

Jochen Hollmann[1], Anders Ardö[2], and Per Stenström[1]

[1] Department of Computer Engineering
Chalmers University of Technology
412 96 Göteborg, Sweden
{joho,pers}@ce.chalmers.se,

[2] Department of Information Technology
Lund University
Box 118
221 00 Lund, Sweden
anders@it.lth.se

Abstract. Latency is a fundamental problem for all distributed systems including digital libraries. To reduce user perceived delays both caching – keeping accessed objects for future use – and prefetching – transferring objects ahead of access time – can be used. In a previous paper we have reported that caching is not worthwhile for digital libraries due to low re-access frequencies.

In this paper we evaluate our previous findings that prefetching can be used instead. To do this we have set up an experimental prefetching proxy which is able to retrieve documents from remote fulltext archives before the user demands them. Using a simple prediction to keep the overhead of unnecessarily transfered data limited, we find that it is possible to cut the user perceived average delay a factor of two.

1 Introduction

Within the last decade, almost all major academic publishers have built up Digital Libraries (DL) which offer fulltext archives of conference and journal articles in electronic form. This has given the research community better accessibility to articles from conferences and journals. Some university libraries have integrated the bibliographic databases from various publishers into a single point of access, so that the users can effectively search many fulltext archives at once. DTV's Article Database Service DADS [1, 2] is such a system which implements a large amalgamated index database covering many publishers, searchable through a gateway which redirects fulltext requests to the publishers fulltext servers. This avoids the cost of huge replicated archives as well as many copyright issues. As a result, digital libraries are realized as a distributed information system.

Transferring documents from remote servers distributed around the world suffers from the latency of networks and servers. While increasing network and

server bandwidth can partly mitigate its impact, the latency time for an unloaded infrastructure is still significant.

Standard techniques to hide latency are *caching* and *prefetching*, and have been applied to computer systems [3] as well as to web-based systems [4]. Caching works by keeping a copy of previously accessed data close to the consumer for future re-use and is thus useful for frequently accessed objects. Unfortunately almost all accessed articles in digital libraries are accessed with very low frequencies. This is the reason why caching does not work well for DL as we observed in a previous study [5].

Prefetching, transferring a data object ahead of access time, can be applied even if the data objects are accessed only once. If perfect knowledge of future accesses would exist, it would be possible to transfer data objects sufficiently in advance so that a local copy would be available close to the user; thus, hiding almost all of the latency. In reality it is impossible to have perfect knowledge about future accesses; hence, a critical issue for prefetching is to establish a good predictor.

Digital article libraries can organize document accesses as a two-step process. First the bibliographic record is shown, then the user can proceed to access the fulltext from the remote server. In a previous study [6] we found that a good prediction can be made if a prefetch is issued when the article abstract is viewed. In this study, the goal is to see whether such a prefetching approach works under real-world conditions by evaluating it in the context of real users.

The specific research questions addressed by this research are: How much of the network and server latency can be hidden? How much does the network traffic increase by superfluous and unnecessary prefetch requests? Finally, how long should we keep prefetched documents in the cache, before we decide that the user will not access them in the future?

To answer these questions we have set up an experimental gateway which implements the proposed prefetching scheme. We have made this system available to all members of our university during almost half a year and logged their activities, as well as the document transfers. By analyzing the resulting user access log files and comparing them to a simulated environment without prefetching, we show the effects of prefetching. We were able to cut the average latency the user experiences by a factor of two. We have also found that this increases the number of document transfers by less than a factor two.

As for the rest of this paper, in Section 2 we describe our digital library system followed by the experimental setup in Section 3, before introducing the evaluation method in the next section. In Section 5 we show our results. Section 6 gives an overview of prefetching mainly in the area of the World Wide Web. Finally we summarize and conclude.

2 Distributed Digital Library for Articles

The aim of every library is to make a large collection of publications easily accessible to the readers. To achieve this, libraries used to build up an archive as well as an index to enhance searching for a particular publication. With the shift to digital libraries, libraries have stopped collecting periodicals and instead provide licenses to directly access remote fulltext archives offered as a service from most publishers.

While many libraries also rely on web interfaces provided by publishers, some have built their own gateways to access these fulltext archives. Figure 1, taken from our previous paper [6], shows the architecture of such an approach. In an academic environment the university library would typically run a gateway service which has access to an index database and the remote fulltext servers provided by publishers. The gateway may have a document cache to serve some document requests locally.

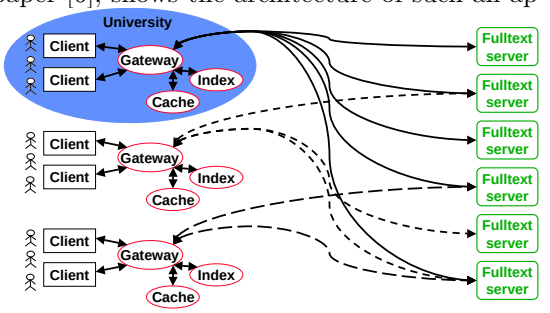

Fig. 1: Architecture of a distributed digital library. Adapted from [6]

2.1 The DADS System

For the study in this paper we have used DTV's Article Database Service DADS [1, 2] as an implementation of the above architecture. DADS was developed with the aim of providing the user with a single point of access to all research related literature by merging the index databases of all major publishers into a single super-database.

The DADS user interface is implemented as a web application. Users interact with the DADS system using a standard web browser, which displays dynamically generated web pages. This implements a user model for the search process shown in Figure 2 (adapted from our previous paper [6]). A user normally starts in the state *search* by providing keywords which should appear within the bibliographic record, for example in the title, the abstract or the authors name. Normally the user wants to reduce the number of hits to a manageable amount. This brings them to the *refine* state. From both states, the user can

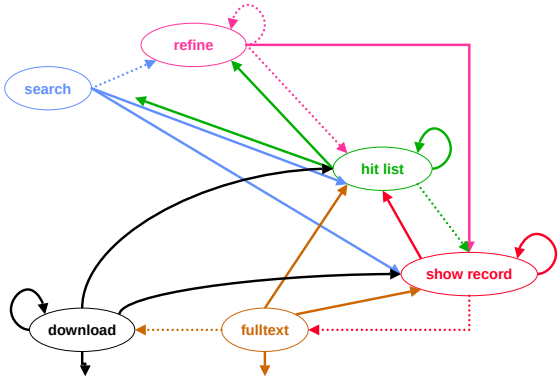

Fig. 2: User model during a search session. Adapted from [6]

move on to the *hit list* state, where up to 10 articles are shown. Selecting one of those articles brings them to the *show record* state, where they can view the bibliographic record including the abstract.

If the users decide that they want to fetch the whole article, they proceed to the *download* state. To do so, the user has to pass through the *fulltext* state, which offers different delivery methods. A paper copy can be ordered from the library for all articles, while online copies in PDF format are only available for a large but limited subset of the indexed articles. A more detailed description and analysis of the user model can be found in our previous paper [6].

2.2 Prefetching

We extended the DADS system by a prefetching component. This was accomplished by adding a file area on the gateway machine as a fulltext cache. We also changed the gateway to manage the prefetch cache in the following way:

- In the state *show record* we check if the corresponding document can be found in the cache. If not, a lock for this document is acquired first before fetching the document from the publishers fulltext archive is started. Locking avoids the occurrence of concurrent prefetches as well as the delivery of incomplete documents to the clients. When the transfer is completed the lock is removed.

- In the state *download* we deliver the paper directly from the cache if it exists and is not locked. If it is locked, we create a dynamic page telling the user that prefetching is in progress and program the user's browser to poll the download page with a short delay again. If it is neither locked nor existing – which is an error condition – we start the actions of the *show record* state and program the user's browser as above.

These two changes are sufficient to implement a simple prefetching strategy.

2.3 The Aim of this Study

Given the above system, the overall question is if it allows most of the latency to be hidden. Furthermore we want to find out how large the overhead in terms of increased network traffic is. Particularly we are interested in the following issues:

- The percentage of documents that can be served by prefetched documents from the cache. This critically depends on the time between viewing the abstract and requesting the document.

- How many of the documents are prefetched, but never accessed by the user? Compared to the useful prefetches this gives a measure for the overhead introduced by our scheme.

- Prefetched documents not picked up by the user can potentially pollute the (limited) cache. To avoid this, there should be a strategy for when a document should be considered useless and evicted from the cache.

3 Experimental Setup

In order to evaluate our method of prefetching we have set up a DADS gateway at Chalmers University of Technology in Gothenburg, Sweden. The gateway was available to all members of the university, approximately 8000 students and 2000 employees. In order to attract as many potential users as possible on campus, the prefetch experiment was announced in the library newsletter to all departments. It was also promoted as a top link on Chalmers libraries front page. Our volunteer users had not used DADS before, but all users had access to the fulltext archives before by using each publishers web interface. We conducted the prefetch experiment from March 1 until August 15, 2002.

We implemented the gateway using a retired commodity PC running Linux on a Pentium processor with 166MHz and 64MB of RAM. The machine had a disk with 1.6GB used for storing the OS, the gateway and the cache for prefetched documents. Our low cost solution did not have the index database available locally. Instead it connected to the remote index database of the original DADS system in Copenhagen, Denmark. While this inevitably slowed down the navigation interface slightly, it was hardly noticeable, because of the good network connection to Copenhagen, Denmark (14 hops, typical round trip time 17ms, measured bandwidth 1.5MBit/sec). It did not have any influence on the timing behavior we wanted to observe, because this is given by the users and not by the gateway.

We did not implement any cleanup strategy for prefetched documents, neither for those delivered to the user nor for those which were never requested. Instead we kept all prefetched documents in the cache. Only once did we have to clean the cache due to the limited disk space; but again this had no influence on the timing behavior for our measurements.

The navigation interface was stripped down to offer only searching (standard DADS does also include browsing). We configured the interface to search only for articles with fulltext available by default, but did not remove the possibility of searching even for articles without an online fulltext copy. We made this decision because we wanted the users to consider the system as useful as possible, in order to attract as many people as we could. Nor did we remove the *fulltext* page (see Figure 2). While we removed the option to order a printout copy from the library, we still gave the users two alternatives. The first was to download the article from our cache, the second to download the article from the publishers web site. This intermediate step did influence the timing behavior of the user, because the user needed to click on one more hyper link, but we think that the effects were in the range of a second, because the user did proceed quickly.

This decision has the following advantages. Firstly the user could compare our prefetching gateway with the publishers server to experience the difference in speed. Secondly, in case our prefetcher had problems, the users could still get the documents from the publishers fulltext server. And in fact these issues came up, mainly due to the access protection strategies used by some publishers.

4 Evaluation Method

In order to analyze the transfer time, start and end points of each document transfer have to be logged. On a per document basis, the prefetch transfer time PTT is defined as the transfer time from the fulltext server to the gateway. Similarly, the client transfer time CTT is defined as the transfer time from the gateway to the client. Note that CTTs are only available for documents requested by the users. Additionally, we define the download time DT, which is the user observed delay. Also, the publisher of each document is available.

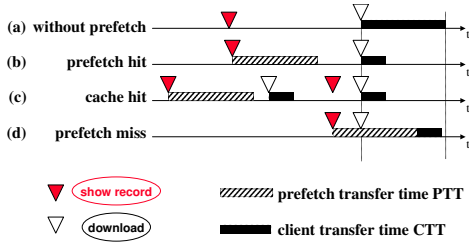

Fig. 3: Occurrence of events

4.1 Classification of Events

For our analysis, we apply the following scheme shown in Figure 3. The download requests from the client are used as time alignment in the following cases:

(a). The figure shows the DT, when fetching directly from the fulltext server. This DT is equal to the PTT in all other cases assuming that the gateway has approximately the same network distance as the clients.

(b). In the case of a successful prefetch, the gateway can prefetch the whole article from the fulltext server before the client requests it. This is possible because the *show record* in the above example occurs long enough in advance. When the client transfers the document from the gateway cache later, the experienced DT will be reduced to CTT due to the local access hiding latency.

(c). Because no cache cleanup strategy is implemented, there may be cases where the document can be served from the cache avoiding another prefetch of the same document. Even in this case the download time is reduced.

(d). In this case our prefetching strategy failed. The time frame from *show record* to the actual *download* is too short to transfer the complete document. In our implementation this delays the transfer to the client. This can potentially lead to an increased DT. In the worst case DT is the sum of PTT and CTT. Streaming the data through the proxy instead of waiting for the prefetch would avoid this, but we decided against it in favor of a simple clean implementation.

Using the above event classification scheme, we can classify all of the download events in our traces into one of the categories (b) to (d) (see Figure 3).

	all publishers	*E* only	*I* only
Prefetch hits:	70.46%	84.17%	66.53%
Prefetch misses:	14.93%	3.89%	15.04%
Cache hits:	14.60%	11.94%	18.51%
Overhead:	78.63%	63.06%	69.41%
not used/used:	530/674	198/314	236/340
Average download time			
with prefetching :	2.21s	1.16s	3.19s
without prefetch :	4.94s	2.13s	7.30s

Fig. 4. Field test prefetch statistics

4.2 Comparison to Non-Prefetching Systems

In order to evaluate our particular configuration it is necessary to compare it to other possible setups. A simulation of alternatives, based on the time stamps available from the traces, is a simple way to do this while preserving the timing behavior of the users. Additionally we need the assumption that the PTT equals the DT without prefetching in place. This is reasonable, because both the client and the gateway are very close to each other compared to the network distance from the fulltext servers.

To simulate a system without prefetching, we move the PTT of case (b) on the time line to the download event, hence transforming it into case (a). This will create a new end point for the DT.

An important part of the analysis is the increase in document transfers, which is caused by prefetched documents not requested by the user later on. We define the overhead introduced by dividing the number of prefetched documents by the number of downloaded documents.

To do this we keep the state on a per document basis. Once a document is prefetched it is useless until it is requested by the user, where it becomes useful.

During our analysis we have found that only two large international publishers were heavily used, while the remaining publishers in our experiment experienced less than 100 download requests each. We have therefore decided to present our results for Publisher *E* and *I* as well as for all publishers together. This is especially interesting as *Es* fulltext archive is approximately 10 times faster than *Is* archive, both in bandwidth and round trip time. (E: 16 ms round trip, 200 Kbytes/s, 2.3 million articles; I: $>$ 120 ms, 1-30 KBytes/s, 1 million articles). The difference mainly comes from the fact that *Es* archive is located close by in Europe, while *Is* archive is located in North America.

5 Results

In the first analysis we want to be convinced that our simple prefetching scheme works. Table 4 shows the results. The most important observation is that we get a considerable amount of prefetch hits. Not surprisingly, the prefetch hit rate is

a lot higher for the fast E source than the slow I server. Obviously the opposite applies for the prefetch misses.

The 10-20% hit rate due to caching is about equal for all three cases and shows what would be achievable by a pure caching strategy. As for the traffic overhead due to prefetching, we introduce a factor of less than two.

The analysis of average download times shows that it is possible to shorten the download time a factor of two consistently in all cases. Considering that this result could be still improved by overlaying the download phase with the prefetching phase for all hits during prefetch, this result is really encouraging. Hence we conclude that one solution to the latency problem in the domain of digital libraries is prefetching.

This leaves us with one question to answer. How long should we keep prefetched documents in the cache before we decide that they are not going to be used?

To answer this question, we have analyzed the time between the *show record* and the *download* state, where available. Figure 5 shows the cumulative distribution during the first hour after an abstract was shown on a logarithmic scale. We have also included our previous findings [6] marked *original DADS*, where the users moved less quickly to the document download. Hence our assumption that the users proceed quickly through the *fulltext* state is sound. Also we can clearly see that in almost all cases it is sufficient to keep the article for an hour. If it is not downloaded by then, we can almost be certain that this prefetch wasted only bandwidth. Note that this figure does not show the timing between subsequent accesses to the same document.

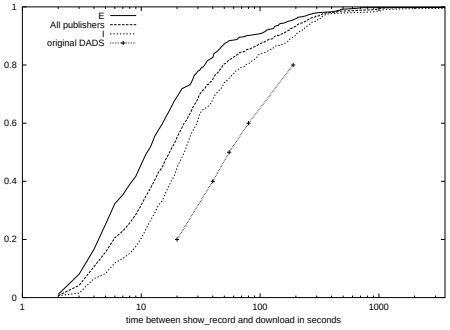

Fig. 5: CDF for timing between viewing the abstract and downloading the document

Hence the conclusion is that we do not have to keep articles for more than one hour, which allows the cache size to be limited.
Considering these results, we should modify our strategy as follows:

- We keep the strategy to fetch an article into the cache, when the user views the abstract.
- If no download occurs during the first hour, we discard the article.
- If a download occurs within an hour, we remove the article with the download.

Note that this approach will eliminate the cache hits, as articles are always removed from the cache on the first download.

6 Related Work

An alternative to prefetching is traditional caching, which not only reduces the latency but also reduces the data transferred. Since in digital libraries the articles are neither modified by the clients nor updated on the servers, this seems to be an ideal approach at first glance. Unfortunately caching requires multiple accesses to the same document in order to work. In a previous study [5] we have shown that this is not the case for 85% of the accessed documents, when not considering re-accesses from the same client. So even with prefetching it does not make sense to keep once accessed articles in the cache for the future. Apart from the small latency gain and the costs, this can also involve copyright issues.

Hence for distributed digital libraries we propose to use document prefetch only, a topic not studied before to the best of our knowledge. The closest area of research is prefetching in the World Wide Web. See Lee [4] for a quick overview of both caching and prefetching and Duchamp[7] for a more in depth overview of the early work done.

Two main categories of different approaches can be found in the literature. The most popular approach is to use proxy or server log analysis in order to predict future accesses of the clients [7–14]. However building up a probabilistic model in one way or the other requires that the candidates for prefetching have been accessed many times before. As we have just seen in the case of caching above, we are missing this property, at least at the university level. Hence, these methods are not applicable to digital libraries.

The second approach in web prefetching is to extract the hyperlinks contained in web documents and prefetch based on those links. As a result, this approach is not limited to pages previously accessed by a particular client or other users.

Eden et al. [15] proposed to let the user mark links, which can be prefetched by the client ahead of time, claiming that this would not increase the amount of data transfered, while still reducing the latency by a factor of three. While extremely simple, it is unrealistic to assume, that the users would be willing to mark all interesting links as well as that they would always follow all of their own hints. Hence, approaches which do not involve the user will be more realistic.

Chinen and Yamaguchi [16] analyzed a web proxy based implementation, where the proxy is placed close to the user in the same way as our gateway. Their proxy uses the first 10 links encountered in the parsed web pages to fetch the linked pages including a maximum of ten images. Their analysis showed in one configuration a 64% hit rate reducing the average latency from 15 to 6 seconds by increasing the traffic by a factor of 2.5.

The work of El-Saddik et al. [17] improved this scheme by ranking the included links according to the words in the link description text, which is compared to the description texts clicked by the user earlier. The underlying reasoning is similar to ours in the way that they try to get information about the intention of the user.

They use a maximum of 15 prefetched web documents per web page and found a lot of generated network traffic: 10 times the amount using the Chinen and Yamaguchi approach and 4.29 times the amount than using their own

approach, in both cases compared to the amount of traffic without prefetching. The more recent work of Davison [18] is quite similar to their approach. Ibrahim [19] also implemented a similar approach using a neural network targeting news web sites. The neural network is used to learn about the interests of a particular user using keywords in the anchor text. They report a low waste rate of 20% to 30%.

WebCompanion [20] by Klemm takes another approach. It selects the prefetched links by the quality of the network connection. Klemm also proposes a name resolutions cache, used to find the web servers.

Cohen and Haim [21] proposed setting up just the network connection to the server in advance instead of prefetching the documents themselves. This is done by doing the name resolution early and establishing the TCP/IP connection. They also force the web servers to be warmed up for the imminent request by asking for the document size. Hence the server has to access the file system. Their techniques are clearly applicable in our case, and could have yielded additional gains. In fact, we know the publisher's servers in advance and could use (multiple) persistent connections to the few fulltext server to speed up transfer times even more.

Foxwell and Menasce [22] studied the effects of prefetching for web search engines. They reported only 14% latency reduction for their best example. This was achieved by prefetching the top eight matching documents of all queries, with a maximum average hit ratio of 35%.

Our prefetching approach is much simpler compared to the above approaches [16–20]. Instead of limiting the potential documents or using a selection scheme to prefetch the highly likely pages, our users give us the information which document is most likely to be accessed for free by moving to the *show record* state. With this information we achieve similar latency reductions with lower overhead, when compared with current web prefetch methods. Typical values are 50% latency reduction with 150% increase in network traffic [17, 20], while we achieved the same reduction with less than 80%.

This navigation scheme is possible due to the structure of having bibliographic information available from the index database corresponding to the fulltext articles. The Web in its current form does not offer this structure, hence it is much harder to build a prefetching system. It is interesting to see that old fashioned ways of organizing information, as in traditional libraries, still have their value in the fast moving digital world.

7 Conclusions

We have set up a prefetching experiment for fulltext servers from major academic publishers. The results from the experiment show that we could reduce the average experienced download time to 1/2 using a simple scheme. At the same time the additional introduced document transfers accounted for less than half of all transfers. We also found that holding articles for a maximum of one hour in the cache is sufficient.

These encouraging results, as compared to web prefetching, are attributable to the well-structured user model digital article libraries provide. By keeping the user "busy" with reading the bibliographic record, including the abstract, we have enough time to prefetch the article in most cases. Because the user has already selected this particular article at that time from a hit list, we experience the extremely high likelihood that they will proceed to download the article – in our case from a prefetch cache close to the user.

Acknowledgments

We are grateful to Prof. Dr. Ulrich Rüde and Dr. Graham Horton from the institution of system simulation at the Friedrich-Alexander-Universität, Erlangen-Nürnberg, Germany for providing us with infrastructure and a magnitude of insights how to gather and analyze relevant data.

We would also like to thank the following publishers for allowing us to prefetch fulltext articles from their fulltext servers: Elsevier, IEEE, ABI, Academic Press, IOP, Springer, Emerald, Kluwer Academic Publishers, RSC and IPG. Special thanks go to Silke Struve and David Ashby for proof-reading.

This research is supported by the NORDUnet2 initiative.

References

1. Ardö, A., Falcoz, F., Nielsen, T., Shanawa, S.B.: Integrating article databases and full text archives into a digital journal collection. In Nikolaou, C., Stephanidis, C., eds.: Proceedings of the Second European Conference on Research and Advanced Technology for Digital Libraries, ECDL'98. Volume 1513 of Lecture Notes in Computer Science., Springer-Verlag (1998) 641–642
2. Sandfær, M., Ardö, A., Falcoz, F., Shanawa, S.: The architecture of DADS - a large digital library of scientific journals. In: Online Information 99, Proceedings. (1999) 217–223
3. Hennessy, J., Patterson, D.: Computer Architecture: A Quantitative Approach. second edn. Morgan Kaufmann Publishers Inc. (1996)
4. Lee, D.C.: Methods for web bandwidth and response time improvement. In Abrams, M., ed.: World Wide Web - Beyond the Basics. Prentice Hall (1998)
5. Hollmann, J., Ardö, A., Stenström, P.: Prospects of caching in a distributed digital library. Technical Report 03-04, Department of Computer Engeneering, Chalmers University of Technology, S-41296 Göteborg, Sweden (2003)
6. Hollmann, J., Ardö, A., Stenström, P.: Empirical observations regarding predictability in user access-behavior in a distributed digital library system. In: Proceedings of the 16th International Parallel and Distributed Processing Symposium, Fort Lauderdale, FL, USA, IEEE (2002) 221–228
7. Duchamp, D.: Prefetching hyperlinks. In: Proceedings of the Second USENIX Sysmposium on Internet Technologies and Systems, Bolder, CO, USA, USENIX (1999) 127–138
8. Padmanabhan, V.N., Mogul, J.C.: Using predictive prefetching to improve world wide web latency. ACM SIGCOMM Computer Communications Review **26** (1996) 22–36

9. Markatos, E.P., Chronaki, C.E.: A top 10 approach for prefetching the web. In: Proceedings of INET'98 Conference, Geneva, Switzerland (1998)
10. Palpanas, T., Mendelzon, A.: Web-prefetch using partial match prediction. In: Proceedings of the 4th International Web Caching Workshop (WCW'99), San Diego, CA, USA (1999)
11. Sarukkai, R.R.: Link prediction and path analysis using markov chains. Computer Networks **33** (2000) 377–386
12. Yang, Q., Zhang, H.H., Li, T.: Mining web logs for prediction models in www caching and prefetching. In: 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining KDD'01. (2001)
13. Wu, Y.H., Chen, A.L.P.: Prediction of web page accesses by proxy server log. World Wide Web **5** (2002) 67–88
14. Chen, X., Zhang, X.: Coordinated data prefetching by utilizing reference information at both proxy and web servers. In: Proceedings of the 2nd ACM Workshop on Performance and Architecture of Web Servers (PAWS-2001), Cambridge, MA (2001)
15. Eden, A.N., Joh, B.W., Mudge, T.: Web latency reduction via client-side prefetching. In: IEEE International Symposium on Performance Analysis of Systems and Software, Austin, TX, USA, IEEE (2000) 193–200
16. Chinen, K., Yamaguchi, S.: An interactive prefetching proxy server for improvement of www latency. In: The Seventh Annual Conference of the Internet Society (INET 97), Kuala Lumpur, Malaysia (1997)
17. El-Saddik, A., Griwodz, C., Steinmetz, R.: Exploiting user behaviour in prefetching www documents. In: Proceedings of 5th International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Service (IDMS98). Volume 1483 of Lecture Notes in Computer Science., Oslo, Norway, Springer-Verlag (1998) 302–311
18. Davison, B.D.: Predicting web actions from html content. In: Hypertext 2002: Proceedings of the Thirteenth ACM Conference on Hypertext and Hypermedia, ACM (2002) 159–168
19. Ibrahim, T., Xu, C.Z.: Neural nets based predictive prefetching to tolerate www latency. In: Proceedings 20th IEEE International Conference on Distributed Computing Systems. (2000) 636–643
20. Klemm, R.: Webcompanion: a friendly client-side web prefetching agent. IEEE Transactions on Knowledge and Data Engineering **11** (1999) 577–594
21. Cohen, E., Kaplan, H.: Prefetching the means for document transfer: A new approach for reducing web latency. In: Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies INFOCOM 2000. Volume 2., IEEE (2000) 854–863
22. Foxwell, H., Menasce, D.A.: Prefetching results of web searches. In: Proceedings of the 1998 24th International Conference for the Resource Management and Performance Evaluation of Enterprise Computing Systems, CMG. Part 2. Volume 2., CMG (1998) 602–609