Empirical Observations Regarding Predictability in User Access-Behavior in a Distributed Digital Library System

Jochen Hollmann
CSE, Chalmers
412 96 Göteborg, Sweden
joho@ce.chalmers.se

Anders Ardö
DTV, DTU
2800 Lyngby, Denmark
and@dtv.dk

Per Stenström
CSE, Chalmers
412 96 Göteborg, Sweden
pers@ce.chalmers.se

Abstract

Today document archives are geographically distributed but often not replicated. This can potentially result in a low quality of service in terms of reduced availability and long user-perceived access times. Instead of indiscriminate replication we study the effectiveness of caching techniques such as prefetching and selective preloading.

Our technique analyzes whether user access behavior is predictable enough to guess what articles to prefetch or to preload based on access logs from DADS, a digital library system for scientific journal articles developed at DTV, the Technical Knowledge Center of Denmark. We have found that once a literature search has been narrowed to up to ten articles, there is a high likelihood that some of them will be eventually downloaded. This suggests that prefetching can be used to hide the article transfer latency. We have also found that 80% of the article downloads are confined to less than 20% of the journals, so preloading a small fraction of the digital library database could significantly shorten the access latency and improve the availability.

1. Introduction

The number of digital library services and users is increasing dramatically with the advent of the Internet. As a result, there is a growing need for a federated service that provides the user with an integrated access to the fulltext archives as well as bibliographic information provided by multiple publishers.

The objective of DTV's Article Database Service (DADS)[1, 14] is to provide the users with an integrated search facility and a direct electronic document delivery of scientific journal articles. The system has been in production since 1998 and serves many, mainly Danish, universities; in particular the Technical University of Denmark, were DTV, the Technical Knowledge Center of Denmark, is located. In contrast to many digital library services from large publishers, used by many libraries today, DADS integrates information from different sources into a single point of access with a generic interface – a one stop shop (for journal articles).

DADS consists of a number of databases encompassing indexing databases and fulltext archives. Users from each organizational unit, for example a university, have access to this collection of databases through a web server with a common gateway interface. Each organizational unit usually connects to DADS through a dedicated gateway.

Implementing digital libraries using a centralized structure poses severe scalability limits. The user-perceived access latency encompasses both the network latency as well as the time to process requests at the various servers involved. With a growing number of users, this latency is increasing owing to contention effects which may reduce the user-perceived quality-of-service. Another scalability issue concerns reduced availability; a centralized digital library structure is prone to single-point-of-failures.

These scalability issues can be addressed using caching techniques. In an ongoing project, we are investigating the prospects of two such techniques: *prefetching* and *preloading*. Prefetching means that data that is anticipated to be accessed in a near future is brought closer to the user before it is actually used. Preloading, on the other hand, means that data that is anticipated to be accessed frequently is cached locally. These techniques have been applied to a spectrum of systems in which it is essential to reduce or hide the latency to access remote data ranging from cache hierarchies in computer systems [7] to web caching [4]. Their use in digital library systems has not been addressed so far to the best of our knowledge.

In this paper, we study the effectiveness of prefetching and preloading. Prefetching effectiveness depends on the accuracy by which future bibliographic requests can be predicted. Using request logs collected during one year, we find that at the time users have narrowed down the search to a handful articles, the likelihood of requesting one of them

Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'02)
1530-2075/02 $17.00 © 2002 IEEE

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: IEEE Xplore. Downloaded on October 20, 2008 at 07:50 from IEEE Xplore. Restrictions apply.

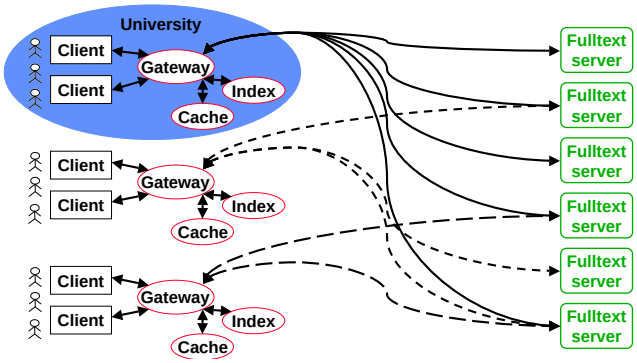

Figure 1. Distributed Digital Library Architecture

from the fulltext archive is quite high. This suggests that prefetching can be used to hide access latencies. Another finding is that preloading only a small fraction (typically 20%) is enough to satisfy more than 80% of the user requests. While a brute-force solution for a digital library would be to preload all fulltext archives locally, the current local fulltext archive of DADS contains around 2.5 million documents with a total size of about 1 Tb – data from only one publisher. Considering how many publishers exist in the world, full replication seems rather expensive.

As for the rest of the paper, we describe a general distributed digital library architecture and the DADS system in particular in more detail in the next section. Then, we present the methodology used to do the analysis of user request logs in Section 3. Section 4 presents our experimental observations along with a discussion about their implications. Finally, we contrast our findings to other research done in this area before we conclude.

2. A Digital Library System

The primary function of a library is to organize published work (information) from different sources as well as to act as an intermediary for finding and accessing the content. Earlier libraries used to build up a physical collection from different publishers on site. If a certain book or article was not available in the local collection, it was typically ordered from another library or the publisher.

With the advent of computers and networks like the internet, publishing information and running a library has changed radically. While searching information has become much easier and quicker, the primary function of a library remains the same. What has changed is the method for distributing the bibliographic and content data. Today many libraries provide an electronic search interface to bibliographic databases and some do also provide direct digital access to the content.

2.1. Digital Library Architecture Overview

Figure 1 shows the schematic architecture of such a digital library. A client application such as a web browser is used in order to connect to the gateway provided by the local library. In an academic context, this is typically a university library. The gateway executes on the user's behalf queries against the index of the library.

It is technically feasible for libraries to have the indices locally in order to provide short access times. This is possible because bibliographic data requires only a small percentage of the total storage capacity needed for the fulltext archive.

Once the user has located an interesting publication, in our case an article, it is naturally to download the publication immediately. Because such an article is in most cases not available locally, it needs to be downloaded from a remote fulltext server. Transferring the article from the remote site may take considerable time which can adversely affect the user-perceived quality of the system.

2.2. The DADS system

The DTV Article Database Service (DADS)[1, 14] is an implementation of parts of the above architecture according to Figure 1. DADS offers its services through a web server which acts as a gateway. Hence a standard web browser is used as the client application. The web server uses server side scripts to communicate with the index databases available. Currently a cache component is not implemented.

In the DADS system, articles are not only located on remote fulltext servers. Additionally, DADS has a large local fulltext archive, which can serve the user quickly. Online articles not found in the local fulltext archive are fetched from a remote fulltext server. In contrast to the general architecture as shown in Figure 1, this is not done by the gateway directly. Instead the gateway sends a HTTP redirection to the client, containing the real location of the document. The client will then fetch the article itself.

The system implements two major modes of usage, browsing and searching. Browsing is much like the traditional browsing in a library. The user looks for a journal and browses the issues of this journal. Searching, on the other hand, means finding articles in a more unstructured way by using the index database.

2.3. The DADS User Model

DADS features a simple user model. Every function is coupled to a web page containing some functionality. This can be either a form to be filled in or a page with hyperlinks, which the user can follow. The web browser will transmit

Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'02)
1530-2075/02 $17.00 © 2002 IEEE

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: IEEE Xplore. Downloaded on October 20, 2008 at 07:50 from IEEE Xplore. Restrictions apply.

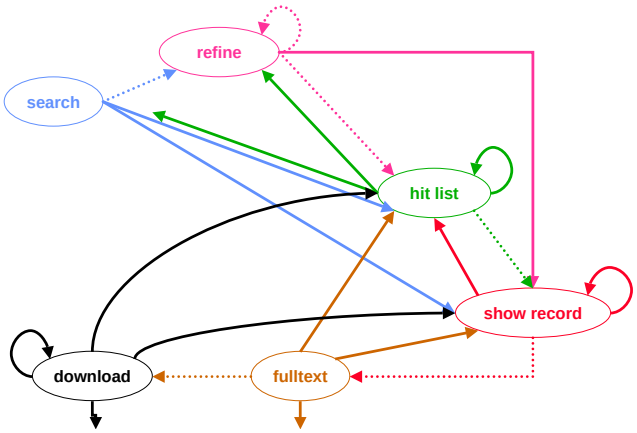

Figure 2. User model during a search session.

the user's decision to the gateway, which will generate the next page.

Figure 2 shows the user/system interaction in a state representation where a user selected action or a system response results in the transition to a new state. We only display the set of states that are relevant for the results of this study.

The user starts from the *search* state, in which the user is asked to enter key words and other information controlling the search. When the outcome of a search results in more than ten articles, which is often the case, the system enters the *refine* state to offer the user an easy way to improve the search.

Then the user can either use this opportunity by giving more search terms, or the user can start to view the hit list of articles that match the search criterion. In this case, the *hit list* state, and its corresponding page, will present the user up to ten hits at a time. The articles are presented with their title, author, etc, but without the abstract.

Once the user selects a particular article the transition to the *show record* state is performed. This is the only state where the abstract together with all other bibliographic data from the index database is shown.

Eventually the user can decide to get the article. The user expects a download to happen immediately when selecting "get article". Because not all articles are available online, the next state *fulltext* does not present the user the article on the screen, but asks the user to take another decision about the delivery method. The system offers photo copying for all articles for a fee and fulltext download for those which have a fulltext available online. If the article is available online, the user can enter the *download* state and download the article.

In Figure 2 we have marked the above described main path with a dotted line.

2.4. Problem Statement and Approach

Ideally, transitions from one state to another would occur in zero time. Unfortunately this is, especially in a distributed system, not the case. Network delays and service times add up and the user perceives delays, which we would like to reduce.

In order to reduce the delays for the end user that downloads articles, we plan to add a proxy cache component, which is located close to the user as shown in Figure 1. The idea is to transfer some articles from the remote fulltext servers ahead of time and store them locally for later use.

Our approach is to use prefetching in order to initiate article downloads early on in the user/system dialogue during a search session as outlined above so that the access latencies can be overlapped with user interaction. Another approach we take is to use preloading to selectively cache a fraction of the fulltext archives that the users tend to access often.

In this paper, we aim at studying the feasibility of both approaches by answering whether (1) prefetching can be triggered sufficiently in advance to hide access latencies and (2) whether a sufficiently small fraction of the fulltext archive will cover most user accesses. We next study the methodology used to answer these questions.

3. Analysis Method

3.1. Logging Infrastructure

Our analysis is based on log files containing user requests captured at a gateway in DADS, where all incoming HTTP requests are recorded by the Apache Web server. By default the log files contain the client's IP-address or hostname, a timestamp for every incoming request, the type and URL of the request as well as the status code and the size of the data delivered. Due to the implementation as dynamic pages, the URL itself contains all the data passed to the backend of the DADS system, for example the state name as described in Section 2.3.

3.2. The Log Files

We base our analysis on the log files gathered from the gateway used primarily by the Technical University of Denmark including its library DTV. For our analysis, we have removed all requests from domains outside the university. We base our studies on the time frame of a whole year from July 2000 to June 2001. The log files occupy 1 GB raw data and correspond roughly to 7 million HTTP requests.

In order to simplify the analysis, we stripped off requests that are irrelevant. The remaining data is stored in a rela-

Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'02)
1530-2075/02 $17.00 © 2002 IEEE

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: IEEE Xplore. Downloaded on October 20, 2008 at 07:50 from IEEE Xplore. Restrictions apply.

tional database. We have also gathered bibliographic data for documents where at least the abstract was viewed once, which is also stored in our database.

After preprocessing we store unique identifiers for the session, the client and the function for each request. Requests from the same host within the same session get a running number in the order of the log file entries to include ordering within a session. Where available, we store a page and document identifier. Finally for each request we store the amount of data transfered and a timestamp, when the request arrived at the server.

The database allows us to quickly retrieve subsets of the requests, for example for a certain session, request type or document. Based on this infrastructure, the analysis is done by counting requests of a certain type, comparing time stamps, etc. More details how the various analyses are done is given in Section 4.

4. Analysis Results and Discussion

In this section we present our analysis results. We first investigate which parts of the system are most commonly used. Then we will identify when prefetching and preloading can be useful.

4.1. State Probabilities

As described in Section 2.3 we associate the user state with the last requested page. Every page type has its own name, which we will also use as the state name of the user. In order to find out the relative frequency of requests of a certain type, we have counted the requests for each type.

The results of our state probability analysis are as follows: Most of the requests are in the article search part. Together with the article viewing part (which is common to browsing and searching), it is responsible for about 50.8% + 22.4% $\approx$ 3/4 of the user activity. In fact, activity due to browsing by journal and article is less than half of the activity due to searching.

We conclude from this statistics that users primarily search for articles. Hence, it seems worthwhile to focus on the searching capability of the system. Interestingly, the impression of librarians we have talked to is that users tend to use the browsing capability more than the searching capability. There are two possible explanations for this observation: (1) Searching requires far more interactions than browsing making the latter more convenient and (2) Librarians primarily talk to users that already know what paper they are looking for and therefore tend to use browsing. Since the techniques we are considering apply to browsing as well, we will not focus on this point subsequently.

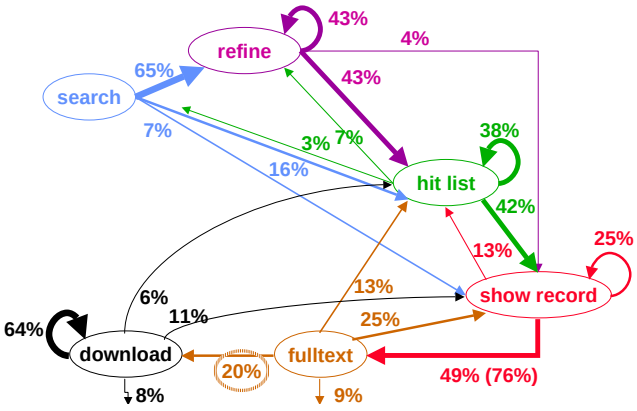

Figure 3. Probability of state transitions.

4.2. Transition Probabilities

Let us now focus on the states involved in the search procedures and article viewing, in particular the most common path across these states. To analyse this, we have selected all requests of a particular type from the database. For all such request, we have then selected the following request within the same session, one at a time. The different types of those requests are counted and weighted against the total amount of request to form transition probabilities. Because users may sometimes open multiple windows which are not traceable in the log files, the statistics we have gathered may contain some uncertainties. Since we believe that such occurrences are rare, it should only slightly affect the statistics.

In Figure 3, we show the state transition probabilities for the interesting paths through the system. The values are rounded to full percentage points. Note that not all possible and existing transitions are shown. This means that we sum of all outgoing transition probabilities does not add up to 100%.

65% of the initial search queries are not precise enough to generate a *hit list* of ten or less articles. As a result, the next state *refine* is used to refine the search within the result set.

When in this state, there is a chance of 43% that the user is going to refine the search again, a chance of 43% that the user decides to see (the first 10 articles of) the result set anyhow and, finally, a chance of 4% that the user did hit exactly one document. In the remaining case, the user goes to some other state.

Normally it appears that the next state of the user is *hit list*, presenting the user with a list of ten or less articles. With a chance of more than 40% the user views an article abstract next. The other possibilities include to scan the continuation of the list (38%) or to refine the search (7%).

Once the user has taken a look at the article record in-

Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'02)
1530-2075/02 $17.00 © 2002 **IEEE**

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: IEEE Xplore. Downloaded on October 20, 2008 at 07:50 from IEEE Xplore. Restrictions apply.

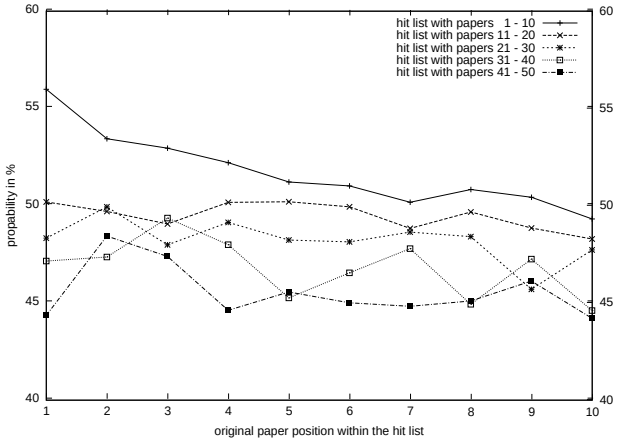

Figure 4. Probability to proceed to fulltext when viewing the bibliographic record including the abstract

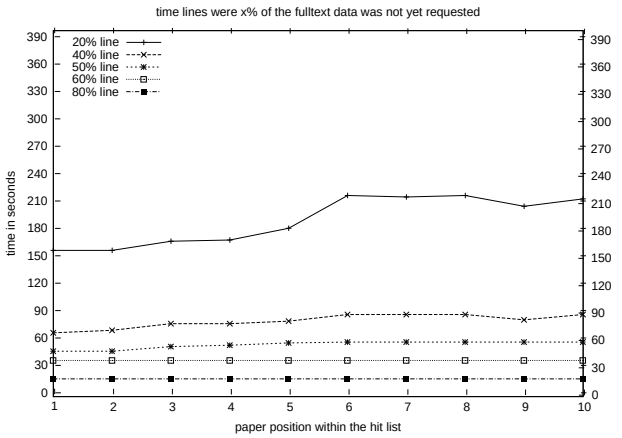

Figure 5. Transfer time from show record to fulltext for the first ten articles.

cluding the abstract, there is a chance of more than 49%, that the article is also requested for viewing. Note that in our system, this does not imply a download directly as not all articles are available electronically.

Interestingly enough, when being in the *show record* state from browsing, the probability is 76% to go to the fulltext state next. This is important, because it shows that the high probability is not limited to searching and also present in the case of browsing.

In 25% of the cases, the user views the next record instead. In 13% of the cases, the user transfers to the next list of articles. Because there is no direct link to this, we assume that the user pressed the back button and continued from an earlier page.

In summary, there is a straight path from searching including a lot of refinement effort to narrow the amount of articles on the hit list. After the refinement, there is a high probability to fetch an abstract and continue to download.

As mentioned, the low probability from *fulltext* to *download* is caused by the articles that are not availble in fulltext, which make up about 80% of the bibliographic records. Intuitively, this percentage will diminish as more articles become available in digital libraries.

4.3. Prefetching Opportunities

For our further discussion we will assume, that the user wants to have the fulltext paper when issuing the fulltext request. Also we narrow our scope and study only on the transition from a *hit list* over *show record* to *fulltext* in more detail.

The obvious technique in this situation would be prefetching. We have a small amount of data, either the articles on the hit list or only one article in case of the abstract being viewed. Also, the time from the *hit list* or *show record* creation to the *fulltext* request is limited. So within this time frame we may or may not be able to prefetch those articles.

For the transition from *hit list* via *show record* to *fulltext* we can actually take into account from which article in the result set the user starts. Our result sets are organized as lists of 10 articles each and articles are numbered 1 to 10 for the position on the list plus the list number itself.

When analyzing timing, we first select all *hit list* or *show record* requests from the database. For all those requests we then try to find the subsequent *show record* or *fulltext* requests. This is the most restrictive analysis possible, because session sequences with intermediate requests are sorted out, which may have increased the decision time.

We ran the timing analysis with precise time stamps collected at the beginning and the end of serving a HTTP request and substracting the service time from the total time delay. Doing this we found no fundamental differences to using only the incomming timestamp, because the service time delay is small compared to the time between the requests.

In the case of the *hit list* to *show record* transition, we compare the total number of *hit list* request with the *show record* requests for a particular paper position to calculate the probability. This is more relaxed than to require that both request follow each other and reflects the possibility that a user actually requests more than one abstract from the same *hit list.* In contrast, the *show record* to *fulltext* transition analysis requires, that both requests follow each other directly.

First we look at the second transition from *show record* to *fulltext*. In Figure 4 we show the result of the probability

Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'02)
1530-2075/02 $17.00 © 2002 IEEE

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: IEEE Xplore. Downloaded on October 20, 2008 at 07:50 from IEEE Xplore. Restrictions apply.

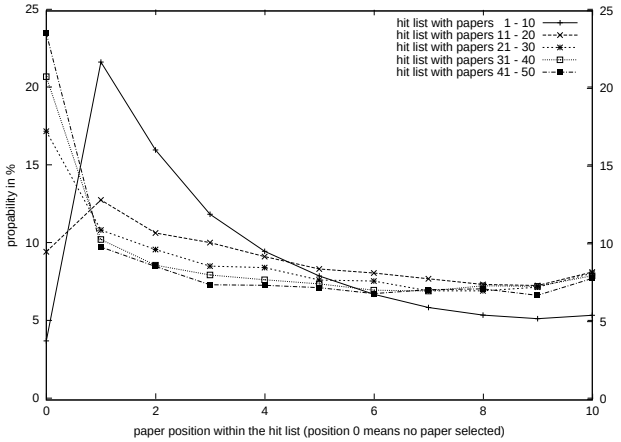

Figure 6. Relative fequencies of bibliographic record/abstract requests compared to viewing a hit list

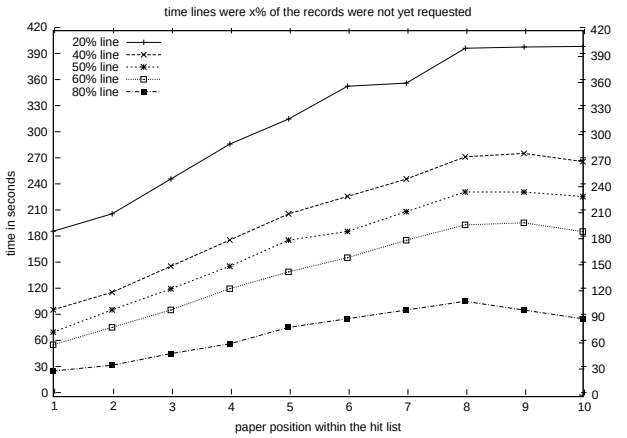

Figure 7. Transfer time from hit list to show record for the first 10 articles

analysis. One can see clearly, that the probabilities are quite uniformly distributed over all list member of the first 5 hit lists in the range 44% to 50%. An exception is the first list of ten articles, where we see slightly higher probabilities especially for the top listed articles.

As for the timing analysis, which was done for the first list of ten articles only, shown in Figure 5, the results are quite similar independent of the papar position. Around 60% of the articles are requested with a think time around 30 seconds.

The timing analysis has shown, that users need around 30 seconds to read the abstract and decide whether to request the article or not. This timeframe seems to be long enough to manage prefetching the article in most cases.

The high probability of around 50% makes prefetching from this state very profitable. By transferring only about the double amount from what is requested, we would be able to satisfy all of the requests, if the time frame of 30 seconds is long enough for a succesfull prefetch.

If the time is not sufficient, we may have to start prefeting earlier, when we have the scope of a list of articles as in the *hit list* state.

Figure 6 shows the probability to take the first, second, etc article from a hit list or none at all. The different lines show the first, second, etc hit list respectively.

We see that the probabilities are almost equally distributed for the various hit lists, with the exception of the first hit list, where the first articles are prefered.

In the timing analysis in Figure 7 – again only shown for the first hit list (of up to 10 articles) – we see a quicker decision for the first articles on the list.

We think that the anomalies on the first list are due to the users assumption, that the results are ranked, which they are not. Jones et al. [8] have found almost twice as high a probability for viewing the first document compared to the second on such a list, no matter if ranked or not. Hence we think that this is not random.

Because the time to decision is shorter for the top entries on the list, and the probability is higher for those items, too, a prefetch strategy should start from the top of the list.

4.4. Preloading Opportunities

The previous analysis focused on user behavior. In this experiment, we change our focus to a system view by analyzing the accesses to our local fulltext archive. An interesting question is whether the accesses are equally scattered over all documents in our fulltext archive or if we can find hot spots. Hot spots could guide us to preload the cache with only parts of the fulltext archive and still achieve a high hit rate within the preloaded data.

As articles can naturally be grouped in several ways, we can study the accesses on various levels, namely on the issue, volume and journal level. By grouping all document accesses on these levels, we can get an estimation on the probability to hit a document locally, if the whole journal, volume or issue would be preloaded.

In this analysis we ignore multiple downloads of the same document (since it mostly represent users issuing a reload request due to impatience). Besides combined with prefetching it assures that all accessed documents will be loaded and available in the cache for repeated downloads.

After sorting the journals according to the number of accesses to different documents within the journals, we can observe hot spots. Figure 8 shows this graph, where the top journal was accessed more than 1500 times. We have done similar studies on the volume and issue level, which show

Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'02)
1530-2075/02 $17.00 © 2002 IEEE

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: IEEE Xplore. Downloaded on October 20, 2008 at 07:50 from IEEE Xplore. Restrictions apply.

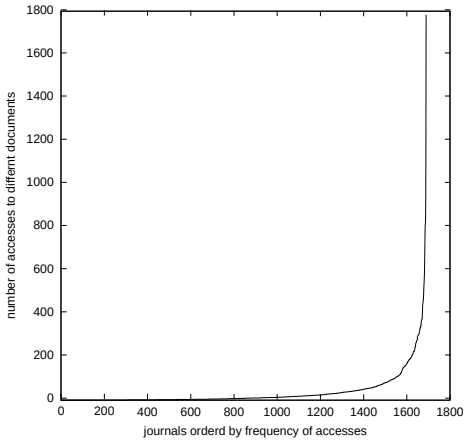

Figure 8. Access to different documents per journal.

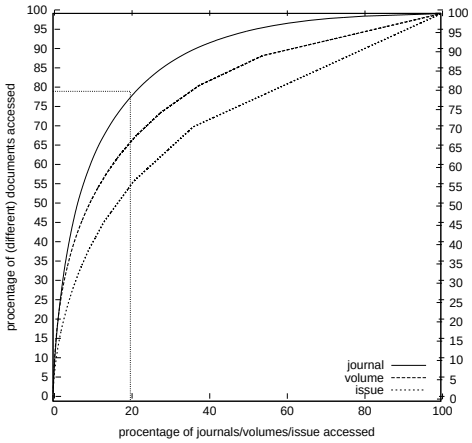

Figure 9. Cumulative distribution function for accesses per journal, volume and issue.

the same distribution, but vary in the number of entities and the top frequencies.

To compare these results, we show them as a cumulative distibution function in Figure 9. The journals/volumes/issues are sorted with decreasing access frequency from left to right. Only those with at least one access were used to scale the axis. For the remaining ones (without accesses), it is unclear, if they were just not found by the users or if they were of no interest to our user community at all. So our graph shows the worst case in this respective. The y-axis shows the coverage of document accesses.

Independent if preloading is done on the issue, volume or journal level, we see that the shape of the curve is aproximately the same. The journal-based preloading is the most effective, but this comes at the cost of a larger granularity; the journal may include many articles which are just preloaded, but never used. This effect is reduced on the finer granularities, but then one has to preload a higher percentage of the volumes or issues.

One effective setup seems to be to preload 20% of the journals, which satisfies 80% of all accesses to documents, as shown by the box in Figure 9.

5. Related Work

Peters [13] gives a comprehensive overview of transaction log analysis in the field of library and information retrieval before digital libraries on the internet became widely used.

[9, 11, 12] have extensivly studied search behavior in network connected digital libraries, but they focus on improving the user interface. Also, their studies are based on collections of computer science papers with its associated user community. In contrast, the digital library we consider is used by researches across most technical disciplines.

Cooper [3] has studied transaction logs of digital libraries to identify their usage and users based on sessions. He and Göker studied a method to separate sessions for web log files [6]. None of the two studied predictablility of user interactions needed for prefetching and preloading.

Related to our work is web caching done with web proxies. An introduction to web cachig can be found in [5], and a bibliography of work done in this area is in [4].

There have been extensive studies how to predict and integrate prefetching into web proxies. Yany et al. [15] suggest a method how to predict future web accesses and show the proxy improvements using logfile traces. Liao and King [10] suggest to integerate caching and prefechting into a common strategy.

All of these studies have a fundamentally different user model inherent in the design of the web. Basically there are no distinct user states, which guide the user interaction as in our system. In addition none of those systems have two decision points for one document access. In contrast, in a digital library the user accesses first the abstract of an article, which gives the system time to prefech only articles with an extremly high probability to be viewed.

An interesting approach to web prefetching is reported by Cohen and Kaplan [2]. These authors have found that a large proportion of the latency perceived by the web users is not attributable to the actual data transfer but rather to the setup times for the download connection. In particular, he mentions name resolution for IP-addresses, setting up the TCP connection and loading the data into the server. By doing theses steps in advance, the authors find a dramatic reduction of latency.

These techniques are not limited to web prefetching and

Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'02)
1530-2075/02 $17.00 © 2002 IEEE

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: IEEE Xplore. Downloaded on October 20, 2008 at 07:50 from IEEE Xplore. Restrictions apply.

can be easily applied in our system at the *hit list* state. Setting up the connections to remote fulltext servers and giving them a hint what document we are potentially interested in, can help to reduce the latency, once the user has decided to view an abstract, when we start to prefetch.

6. Conclusions

A basic functionality of a digital library is to assist the user to find relevant information. One way is to narrow the scope (searching and refining) to a subset of the archive which is viewable by the user. We have shown, that we can use this functionality to reduce the potential candidates for prefetching effectively. In the extreme case of a result set of only one article, combined with the time the user normally needs to read the abstract of this article, we can apply prefetching with a bandwidth overhead of only a factor of two. If the network latency and capacity is good enough, we will be able to fetch the article in most cases before the user actually request it, because the typical user needs enough time to scan through the abstract of an article.

If the network connection is not good enough, we still have two options. We can start to prefech earlier, when the result set is still in the range of 2 to 10 articles. Both parallel and sequential prefetching is possible for the articles on the hit list. Prefetching from this state can have severe implications for both the data amount transfered and the server load. There is also a high probability that none of the data is used, so we may easily see an overhead of a factor of ten.

The second option is preloading of articles, which are located in frequently accessed journals, volumes or issues. By preloading, for example 20% of the journals, we can satisfy 80% of all download requests from the cache. Hence we could reduce the number of remote accesses to 20%.

A combination of these techniques seems to be very valuable. The disadvantage of the early stage prefetching with many article transfers in a short time will be relaxed by the fact that many articles will already be present in the cache through preloading. On the other hand, the preloading strategy benefits from early prefetching, because this handles the 20% of journals not available from the cache.

Preloading may also help increasing the availablility of the system in case of failures in the network or the servers.

Acknowledgment

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 extract analysis relevant data.

This research is supported by the NORDUnet2 initiative.

References

[1] A. Ardö, F. Falcoz, T. Nielsen, and S. B. Shanawa. Integrating article databases and full text archives into a digital journal collection. In *Proc of the Second European Conference on Research and Advanced Technology for Digital Libraries, ECDL'98*, volume 1513 of *Lecture Notes in Computer Science*, pages 641–642, Sept. 1998.

[2] E. Cohen and H. Kaplan. Prefetching the means for document transfer: A new approach for reducing web latency. In *INFOCOM (2)*, pages 854–863, 2000.

[3] M. D. Cooper. Usage patterns of a web-based library catalog. *Journal of the American Society for Information Science and Technology*, 52(2):137–148, 2001.

[4] B. D. Davison. Brian davison's web-caching bibliography. http://citeseer.nj.nec.com/281347.html.

[5] B. D. Davison. A web caching primer. *IEEE Internet Computing*, 5(4):38–45, aug 2001.

[6] D. He and A. Gker. Detecting session boundaries from web user logs. In *Proceedings of the BCS-IRSG 22nd Annual Colloquium on Information Retrieval Research*, Apr. 2000.

[7] J. Hennessy and D. Patterson. *Computer Architecture: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., second edition, 1996.

[8] S. Jones, S. J. Cunningham, and R. J. McNab. Usage analysis of a digital library. In *ACM DL*, pages 293–294, 1998.

[9] S. Jones, S. J. Cunningham, R. J. McNab, and S. J. Boddie. A transaction log analysis of a digital library. *International Journal on Digital Libraries*, 3(2):152–169, 2000.

[10] W.-K. Liao and C.-T. King. Proxy prefetch and prefix caching. In *Proceeding of the 2001 International Conference on Parallel Processing*, pages 95 – 102, sep 2001.

[11] M. Mahoui and S. J. Cunningham. A comparative transaction log analysis of two computing collections. In *4th European Conference on Research and Advanced Technology for Digital Libraries, ECDL'2000*, volume 1923 of *Lecture Notes in Computer Science*, pages 418–423. Springer Verlag, Nov. 2000.

[12] M. Mahoui and S. J. Cunningham. Search behavior in a research-oriented digital library. In *5th European Conference on Research and Advanced Technology for Digital Libraries, ECDL'2001*, volume 2163 of *Lecture Notes in Computer Science*, pages 13–24. Springer Verlag, Nov. 2000.

[13] T. Peters. The history and development of transaction log analysis. *Library Hi Tech*, 42(11):41–66, 1993.

[14] M. Sandfær, A. Ardö, F. Falcoz, and S. Shanawa. The architecture of DADS - a large digital library of scientific journals. In *Online Information 99, Proceedings*, pages 217–223, Dec. 1999.

[15] Q. Yang, H. H. Zhang, and T. Li. 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*, aug 2001.

Proceedings of the International Parallel and Distributed Processing Symposium (IPDPS'02)
1530-2075/02 $17.00 © 2002 **IEEE**

IEEE COMPUTER SOCIETY

Authorized licensed use limited to: IEEE Xplore. Downloaded on October 20, 2008 at 07:50 from IEEE Xplore. Restrictions apply.