Structured Partially Caching Proxies for Mixed Media

Frank T. Johnsen, Carsten Griwodz, and Pål Halvorsen

Department of Informatics,
University of Oslo,
{frankjo,griff,paalh}@ifi.uio.no

Abstract. News on demand features user interaction, interdependent media and is used by different client types. These requirements are not yet accommodated by a single solution. We address this issue by utilizing a priori knowledge about news articles' structure encoded in a presentation plan. Using this, we investigate the gains that knowledge about structure can have for a caching proxy. Our simulations show that *structured partial caching* of media object segments achieves a higher byte hit ratio compared to a non-structured caching scheme when access patterns follow a Multi-Selection Zipf distribution.

1 Introduction

In recent years, we have seen an increase in services available on the Internet. More people peruse the Internet for information than ever before, and replace their modems with more capable ADSL lines. Service providers respond to this trend by providing even richer media for their customers. Some broadband providers now offer their customers true video on demand (VoD), a service one earlier could only dream of when using a modem. The next trend, which we anticipate will be even more popular than VoD, is news on demand (NoD). VoD requires a certain amount of bandwidth for a single stream, but is quite predictable in behavior. A user may perform simple VCR-like operations on a movie, such as playing, stopping or skipping. NoD, on the other hand, has interrelated files of different kind, including continuous and discrete media, and can offer more advanced interactivity to the clients. The clients themselves can have varying capabilities, and range from powerful PCs to PDAs and mobile phones. Also, the flexibility introduced by utilizing mixed media for NoD brings new requirements to proxies in distribution networks. We explore the concept of mixed media below when describing our NoD scenario.

A lot of research has been committed to VoD, and there exist several techniques which can be applied by the providers in order to make effective use of the resources in the network, such as caching proxies. Caching is a means to lower traffic at the origin server and save bandwidth between server and proxy, and it is our belief that efficient caching is a key to any content distribution scheme as it can lower the overall costs. In order to accommodate the special

C.-H. Chi, M. van Steen, and C. Wills (Eds.): WCW 2004, LNCS 3293, pp. 144–153, 2004.

needs of NoD, we have devised our own variant of partial caching [1], which we call *structured partial caching*. Our scheme is specially tailored to make use of the structure inherent in mixed media, which probably will be utilized in the future for NoD. Our goal is to make a partial caching scheme that is specialized for mixed media content in a NoD scenario. To investigate possible gains we have written a simulator.[1] The results are promising, showing that using knowledge of structure can significantly increase byte hit ratio under certain conditions.

The rest of this paper is organized as follows: In section 2 we present the NoD scenario in detail. The differences between NoD and VoD are discussed. Further, important issues like structure and mixed media are defined. Some of the research done on VoD can be applied in a NoD scenario as well. We point out some of these important ideas in section 3 on related work. Our structured partial caching scheme is described in section 4, where we explain how knowledge of structure can be utilized in order to perform efficient caching of mixed media data in our NoD scenario. Details concerning the simulations are discussed in section 5. Our conclusion and summarized results are given in section 6 together with an outline of future work to be performed.

2 NoD Scenario

In recent years, much research has been performed in the area of VoD. Usage patterns have been investigated [2], streaming issues have been addressed [3] and servers optimized [4]. Also, distribution schemes have been devised and multimedia proxies [1,5,6] added to further enhance performance. However, there are other areas utilizing different media types that are similar to, but less explored than VoD. One such area is NoD. Important aspects of NoD is user interaction, a concept of structure through the use of mixed media content, and a vast array of different clients with different capabilities – ranging from powerful PCs to mobile phones – wanting to use the content. Since the characteristics of NoD differ significantly from those of VoD, old VoD distribution schemes cannot be used directly [7]. We need a system which heeds the needs of NoD.

VoD and NoD differ in a number of ways: VoD access follows the Zipf distribution, and popularity will often remain stable for a long period of time. Further, a video is one linear file, and the access pattern is totally independent of the other videos. NoD, on the other hand, is accessed as described by the Multi-Selection Zipf distribution [7]: Client requests tend to be for groups of articles rather than single items as in VoD. Rather than following a Zipf-distribution for all articles, the news items will be requested in groups, and these groups contain several articles. The popularity span is also much shorter, as an article is usually most popular during its first three days after publication [7]. When speaking of NoD, we assume that the structure of the media objects of news stories are described by some sort of meta data, in the following referred to as

[1] This work is part of the INSTANCE II project (147426/431) funded by the Norwegian Research Council.

a presentation plan. Such a plan or news item encompasses mixed media, for example text, video and audio objects.

We define two types of structure; *internal structure* and *external structure*. The presentation plan gives external structure to the media objects. This structure imposes certain restrictions on the objects, for example saying that two objects should always be retrieved together, after one another, or as an alternative to one another. There are other possibilities as well; a total of 13 were identified in [8]. Knowledge of external structure is important when servers and proxies perform rough synchronization between objects, as irregularities here will affect the user's perception [9] of the media. In addition to an external structure, media objects can also have an internal structure. This is the structure that is inherent in the media object's format, for example through the layering of video.

Mixed media has many similarities to hypermedia [8]; both have different media objects and presentation plans. However, the two differ on some key issues: Hypermedia is strongly focused on navigation issues, whereas mixed media focuses more on streaming and concurrent presentation of different media. Different objects have different spatial and temporal requirements, and objects with a temporal dimension (audio and video) can be streamed, while objects with only spatial requirements must be retrieved all at once. Further, we have some general knowledge about client interactivity and access patterns, as client requests tend to be for groups of articles [7] rather than single items as in VoD. Rather than following a Zipf-distribution for all articles, the news items will be requested in groups, and these groups contain several articles and are accessed following the Multi-Selection Zipf distribution.

In online newspapers, news presentations are usually constructed both from brand new material and from archives. We envision a scenario where news can be made using new media objects, whole or parts of old objects and presentations. This differs from HyNoDe [10], where one would not have reused the presentation plan, only a lot of raw material. In particular, HyNoDe would reuse only subsequences of audio or video clips. We assume a fair amount of content reuse in NoD, including connections between presentations. Using a presentation plan format that supports inclusion, like MHEG [11], might lead to the utilization of sub-presentations, while another presentation plan encoding like SMIL [12] would allow the use of files that are only logically related.

3 Related Work

Existing caching schemes such as prefix caching [6] for streaming media and the full object caching performed by many web proxies such as SQUID [13], though effective for their purposes, can be improved on to better suit a NoD scenario. This is due to the fact that rather than always requesting an entire object, or an object from start to finish, we may also have subset access. The most important issue here is that partial caching [1] should be employed, of which there are currently two main types: You can have partial caching in the time domain, an

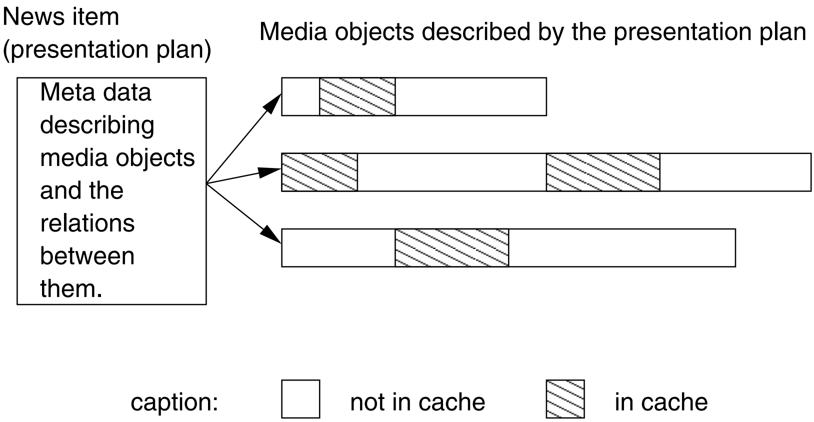
News item
(presentation plan)
Media objects described by the presentation plan
Meta data describing media objects and the relations between them.
caption:
not in cache
in cache

Fig. 1. An example showing a news item and its media objects.

example would be prefix caching [6], and in the quality domain; examples here are the layering of video [5] and the rescaling of video [1].

Our idea is different from existing work in that we are considering the needs of mixed media for NoD. To make even better caching decisions we believe that one should also look at the dependencies between media objects, i.e., the structure of a news story. This is what we call *structured partial caching.*

4 Structured Partial Caching

Figure 1 illustrates a news item and the three media objects it contains. The media objects and the presentation plan itself are all located at the origin server. When a user requests a certain plan, its client software will parse the plan and offer the options it enables to the user. Based on the user's interactions the news item may be downloaded and watched, either in part or in full. The figure shows a case where the plan uses only a few parts of existing media objects, so although the entire objects are present at the origin server, there is no need to cache more than is actually used. The figure illustrates how the proxy performs structured partial caching. Knowledge of internal structure is utilized here. However, the fact that the three media objects are part of the same presentation also imposes external structure on them, which implies that they will likely be retrieved together. This is important when performing rough synchronization between objects, and can be of use to the caching scheme. A caching scheme may guess that such a relation exists, but the synchronization mechanics must know.

We want to investigate if a proxy utilizing knowledge about structure will be more efficient than a proxy that does not, as the more information a proxy has about its cache elements, the more likely it is to make a good decision regarding cache replacement. There are several different ways to measure the efficiency of caching algorithms, but when speaking of efficiency here we mean the *byte hit ratio*. The byte hit ratio is the percentage of requested bytes that are served from the cache [14]. A high byte hit ratio means that there is an increase in saved bandwidth between server and proxy, an important issue when wanting to keep overall transmission costs low. However, making use of as much knowledge as possible about the objects eligible for caching will naturally be more CPU intensive than simpler existing caching schemes. The initial costs of deployment will therefore be higher, as we need more powerful machines for proxy servers. On the other hand, we save bandwidth.

When performing structured partial caching in the NoD scenario we have some extra knowledge about the media objects we are caching that can be used: The structure is known a priori, i.e., we have the presentation plan. Also, there may be a certain reuse of old elements, when old news items are linked and related to current events.

We propose that different media objects should be treated differently by the proxy, and that the meta data neccessary for this should be available in the presentation plans. More precisely, both the external and the internal structure should be described by the meta data. This enables the proxy to handle all content based on meta data, and does not render it useless in the case of a new media format becoming popular. Also, it saves processing power, since the proxy does not have to analyze the bulk of data passing through it, only the corresponding meta data.

If an object can be partially cached and the part that is stored at the proxy is usable in its own right, then the object has internal structure. An example here is layered video [5], where one can choose to cache only lower layers (at the expense of quality). Another example of internal structure is a progressive JPEG picture; here prefix caching of such a picture would render the prefix usable in its own right, rather than if one had the prefix of a picture without internal structure (gif, for example) where the prefix would be useless without the rest.

Our scheme differentiates between objects with and without internal structure in the following manner: Regular prefix caching is used if there is no internal structure. Structured partial caching is used in the cases where there is internal structure. For scalable, linear monomedia formats, this is the same as quality partial caching. In the case where one presentation plan uses the whole object, and clients request this, then the whole object is cached. But, if at some later point another presentation plan reuses only part of previous said object, then it has gained internal structure which the proxy has become aware of, and structured partial caching will be employed henceforth. This is our novel concept, and different from all other caching schemes. The proxy can "learn" about structure based on this new usage of the old material.

Mixed media consists of several different media objects that are structured by meta data in a presentation plan, which describes the relationships between the objects. For our work on structured partial caching, we do not choose a specific encoding format. Rather, we use our own, generic scheme in the simulations so

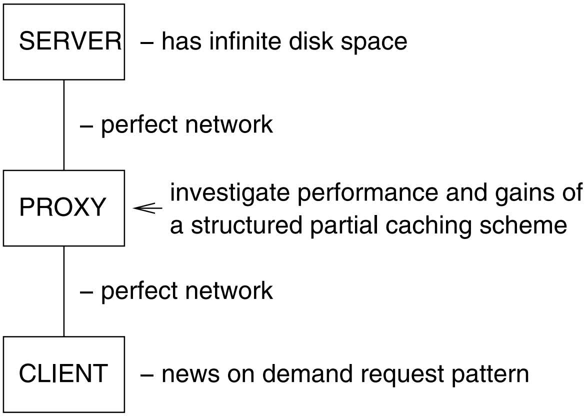
SERVER
– has infinite disk space
– perfect network
PROXY
investigate performance and gains of a structured partial caching scheme
– perfect network
CLIENT
– news on demand request pattern

Fig. 2. The simulator setup.

that the results will be applicable to whichever encoding format one may choose. Because of this, we limit the extent of external structure to a level that can be expressed by all the previously mentioned formats, i.e., we omit the concept of sub-presentations.

5 Simulations

To test our assumption that structured partial caching is indeed beneficial, we have simulated a proxy utilizing this technique. The simulator was implemented in C++ using the lagged fibonacci random generator from the boost library from "http://www.boost.org". The main parts are as Figure 2 illustrates. Client requests follow the Multi-Selection Zipf distribution [7]. The network is assumed to exhibit perfect behaviour, that is having zero delay, no packet loss and unlimited bandwidth. We want to investigate proxy internals only as we seek to determine if structured partial caching is beneficial or not. Thus, we do not need the added complexity of network simulations. The proxy gathers statistics about media object use and performs caching. We let the various modules communicate with each other through function calls, and collect statistics regarding cache replacement and algorithm bookkeeping only.

The news items, which are situated at the server, have been generated using very simple meta data. The data contains information about the number of media objects in the presentation and their attributes, such as size and internal structure. The news items may contain text, audio, video and pictures, and these objects may or may not have internal structure.

The algorithms tested were *LRU over entire objects* and *LRU over parts of objects*. The first of these was implemented as a reference only to see how well a scheme without special measures compares to our enhanced scheme. Caching

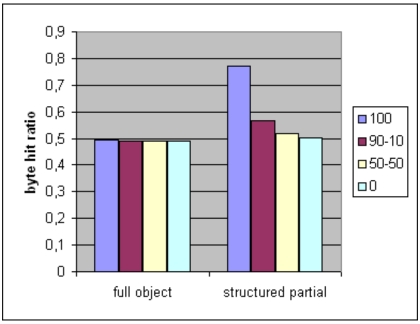

Fig. 3. Large cache, differing quality requests.

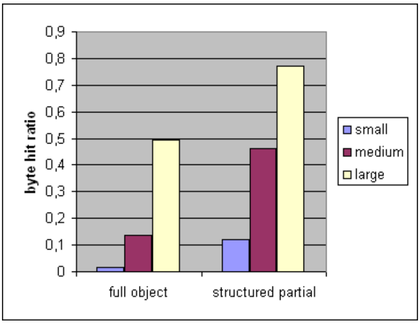

Fig. 4. Varying cache size, low quality requests only.

over parts of objects is partial caching in the quality domain using knowledge of structure.

For all algorithms, we ran tests both with and without an admission policy. The policy used in those cases was that presentation plans should be no older than three days for their objects to be eligible for caching. This was done to incorporate the knowledge that an article usually is most popular in the first three days after its release [7], so that if an older article is referenced it should not displace some of the newer ones in the cache. Old objects that are used in part or in full in new plans will be eligible for caching. Since all or part of an old object can become popular in this way, it is important to perform admission based on the presentation plan's age rather than object age.

We ran simulations using a total of 100000 presentation plans and 270000 media objects, for an average of 2.7 objects per plan. Simulations were performed with three different cache sizes. The small cache has room for an average of 200 whole objects, the medium cache 2000 and the large cache has room for 20000 objects. We distinguish clients that retrieve low quality from those that retrieve high quality objects. By low quality we mean a request for 10% of the media object's data, since progressive quality compression requires the client to retrieve 5–10% to recognize most of the features [15]. Figure 3 shows simulations for the largest cache size and clients exhibiting different quality requests. Clients browsing content have different capabilities. To reflect on this we ran several request combinations, namely such that all of the requests were for low quality content, 90% of the requests were low quality content, 50% of the requests were for low quality content, and all full quality. This is indicated in the figure by the numbers 100, 90–10, 50–50, and 0, respectively.

If only low quality is requested, we see significant gains as cache size increases. This is illustrated in Figure 4. We see that the scheme performs best when all clients request the lowest quality possible from the cache. This is due to the fact that the segments stored in the cache will be quite small, thus enabling many

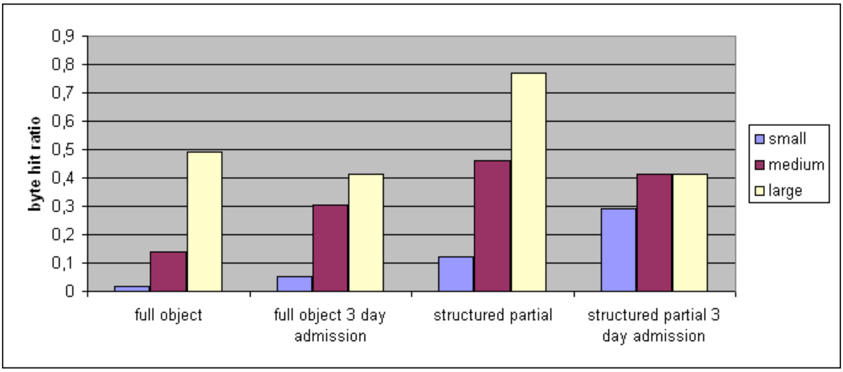

Fig. 5. Varying cache size, low quality requests and use of no vs 3 day admission policy.

of them to be stored compared to a non-partial caching scheme. A mixture of clients requiring various quality levels causes the scheme to perform worse, as is the case for our 90–10 and 50–50 low/high quality requests. Few high quality requests have a big impact on byte hit ratio. When all requests are for high quality, i.e., entire objects, the performance exhibited is slightly better than a non-structured scheme. This arises from the reuse of subsets of objects (we have a reuse probability of 10% in our simulations), where the structured scheme will save only the parts in use, as we saw in figure 1.

The different algorithms have varying CPU demands, and naturally an increase in complexity requires more processing power. The structured partial caching requires 5 times the processing power of whole object LRU when the requests are for low quality only, i.e., many segments in cache leading to much bookkeeping. When requests are for high quality only the processing power required diminishes to 3.5 times that of whole object LRU. We also tried a variant where we pinned all objects belonging to the same presentation plan as the object that was being requested. That variant requires 16 times the processing power for low quality requests and 5 times the processing power for high quality requests when compared to full object LRU. In our simulations there were no additional gains from performing this pinning, as results were identical to those of the structured partial caching which did not perform pinning.

Figure 5 illustrates the difference in byte hit ratio exhibited by the algorithms with no admission policy and our three day admission policy respectively. An interesting aspect to note is that for the smallest cache size the admission policy enhances performance by increasing the byte hit ratio. For the bigger caches, though, it can significantly lower performance by not allowing the cache to store older news as well.

Our results show that for our algorithm of choice, LRU, knowledge of structure has an impact on byte hit ratio. The scheme performs best when there are many requests for parts of objects, i.e., not full object requests. If we assume that the news service will be available for a large variety of clients such as PCs, PDAs and mobile phones, then there is obviously a need for different quality levels for the different clients, and structured partial caching will be particularly useful. For our mixed media NoD scenario the structured partial caching scheme will at all times exhibit a byte hit ratio equal to or better than the scheme not considering structure.

6 Conclusion

In this paper, we have investigated the usefulness of structured partial caching of mixed media content in a NoD scenario. We conclude that the proxies should utilize knowledge about structure provided that there is a mixture of client types. Structure expresses both relationships between media objects (external structure) as well as structure within objects (internal structure). Knowledge of structure can be extracted from the presentation plan, i.e., the meta data which describes each news item. We have investigated the benefits of utilizing knowledge of structure, and our results show that it will enhance a proxy's treatment of mixed media data.

What we have now is an extension of quality partial caching [1], in the future we want to look at extending this to incorporate a means of temporal partial caching [1] as well. Additionally, we want to investigate the additional gains possible by introducing cooperative caching [14]. It should be possible to save further on bandwidth costs when proxies cooperate by exchanging content rather than going all the way to the origin server.

Another important aspect of our simulations is the clients' interactions with the presentations. We want to investigate client patterns in order to refine that part of our simulations. The patterns described in [7] are not particularly fine grained. They are, however, the most detailed ones available at the time of writing, and we did use the knowledge therein for the evaluation of the structured partial caching scheme. At present we are in contact with various newscasters in order to try and obtain logs or other means of statistics that would enable us to pinpoint more realistic NoD client access patterns.

References

1. Peter Schojer, Laszlo Bözörmenyi, Hermann Hellwagner, Bernhard Penz, and Stefan Podlipnig. Architecture of a quality based intelligent proxy (QBIX) for MPEG-4 videos. In *The Twelfth International World Wide Web Conference.*, pages 394–402. ACM, 2003.
2. Carsten Griwodz, Michael Bär, and Lars C. Wolf. Long-term movie popularity in video-on-demand systems. In *Proceedings of the ACM International Multimedia Conference (ACM MM)*, pages 340–357, Seattle, WA, USA, November 1997.
3. Michael Zink, Carsten Griwodz, and Ralf Steinmetz. KOM player - a platform for experimental vod research. In *IEEE Symposium on Computers and Communications (ISCC)*, pages 370–375, July 2001.

4. Pål Halvorsen. *Improving I/O Performance of Multimedia Servers.* PhD thesis, Department of Informatics, University of Oslo, Norway, September 2001. Published by Unipub forlag, ISSN 1501-7710 (No. 161).
5. Reza Rejaie, Haobo Yu, Mark Handley, and Deborah Estrin. Multimedia proxy caching for quality adaptive streaming applications in the Internet. In *Proceedings of the Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 980–989, Tel-Aviv, Israel, March 2000.
6. Subhabrata Sen, Jennifer Rexford, and Don Towsley. Proxy Prefix Caching for Multimedia Streams. In *Proceedings of the Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, pages 1310–1319, New York, NY, USA, March 1999. IEEE Press.
7. Y.-J. Kim, T. U. Choi, K. O. Jung, Y. K. Kang, S. H. Park, and Ki-Dong Chung. Clustered multi-media NOD: Popularity-based article prefetching and placement. In *IEEE Symposium on Mass Storage Systems*, pages 194–202, 1999.
8. Lynda Hardman, Dick C. A. Bulterman, and Guido van Rossum. The amsterdam hypermedia model: adding time and context to the Dexter model. *Commun. ACM*, 37(2):50–62, 1994.
9. Carsten Griwodz, Michael Liepert, Abdulmotaleb El Saddik, Giwon On, Michael Zink, and Ralf Steinmetz. Perceived Consistency. In *Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications*, June 2001.
10. Carsten Griwodz, Michael Liepert, and The Hynode Consortium. Personalised News on Demand: The HyNoDe Server. In *Proc. of the 4th International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS'97)*, volume 1309, pages 241–250. Springer-Verlag, Berlin, Heidelberg, New York, September 1997.
11. L. Rutledge, J. van Ossenbruggen, L. Hardman, and D. Bulterman. Cooperative use of MHEG and HyTime in hypermedia environments, 1997.
12. Lloyd Rutledge, Lynda Hardman, and Jacco van Ossenbruggen. Evaluating SMIL: three user case studies. In *Proceedings of the seventh ACM international conference on Multimedia (Part 2)*, pages 171–174. ACM Press, 1999.
13. Squid web proxy cache. http://www.squid-cache.org/.
14. Michael Rabinovich and Oliver Spatscheck. *Web caching and replication.* Addison Wesley, 2002.
15. David Salomon. *Data Compression - The complete reference.* Springer-Verlag, 2 edition, 2000.