DATA PREFETCHING FOR SMOOTH NAVIGATION OF LARGE SCALE JPEG 2000 IMAGES

Antonin Descampe^{1,a}, *Jihong Ou*^{2}, *Philippe Chevalier*^{3} *and Benoit Macq*^{1}

^{1}Communications and Remote Sensing Lab, Université catholique de Louvain, Belgium
{descampe,macq}@tele.ucl.ac.be
^{2}NUS Business School, Department of Decision Sciences, Singapore
^{3}CORE and IAG, Université catholique de Louvain, Belgium

ABSTRACT

Remote access to large scale images arouses a growing interest in fields such as Medical Imagery or Remote Sensing. This raises the need for algorithms guaranteeing navigation smoothness while minimizing the network resources used. In this paper, we present a model taking advantage of the JPEG 2000 scalability combined with a prefetching policy. The model uses the last user action to efficiently manage the cache and to prefetch the most probable data to be used next. Three different network configurations are considered. In each case, comparison with two more classic policies shows the improvement brought by our approach.

1. INTRODUCTION

Development and diversification of computer networks as well as availability of devices able to produce very high resolution images have created the potential for a new imaging application: the remote access to large scale images. Two examples from different fields are given in [1, 2]. However browsing large scale images over best-effort network is particularly challenging because minimizing the used bandwidth and increasing the navigation smoothness are antagonistic goals. To achieve them both, at least two research areas can be explored.

First, large scale images need to be efficiently compressed, structured and stored. As these images could be as large as several gigabytes, the user accessing a remote server wants to be able to browse various parts of an image, at various resolutions, without having to download the entire file locally. These constraints have highlighted several shortcomings in actual image compression standards, such as JPEG. The lack of resolution or quality scalability is clearly one of the most significant drawbacks. The new image compression standard JPEG 2000 [3] enables such scalability: according to the available bandwidth, computing power and memory resources, different resolutions and quality levels can be extracted from a single bit-stream.

Second, additional techniques could be developed to efficiently manage user requests and downloaded data. One approach to improve the smoothness of the navigation is to set up a cache at the client site and, when the viewer is looking at the just arrived display, transmit to cache the data that could be needed for the next display. The idea is to prefetch data in anticipation of future use as the viewer moves through the image. To make the approach efficient, the issue is to choose which data should be prefetched given the current display the viewer is looking at and past navigation actions the viewer has taken.

In this paper, we develop an analytic modeling framework to assist systematical selection of the right data to prefetch at the right time. The objective is to minimize the wait the viewer experiences and the bandwidth waste when some prefetched data is not used. Thanks to this user's behavior prediction and to the fine scalability offered by JPEG 2000, we achieve a smoother navigation for the user.

Regarding the storage and compression of remote images, numerous contributions had already been made before the development of JPEG 2000 [1, 4]. Their main ideas (multiresolution, SNR scalability, ...) have been grouped together in the JPEG 2000 framework and one of the 13 parts of the standard is even dedicated to an interactive protocol for accessing remote images [5]. Static web caching has been deployed since the late nineties and there is therefore a large body of research literature on what web pages to cache and how to organize the cache. Among these contributions, caching strategies adapted for images have been developed, as in [6]. Nevertheless, data prefetching based on dynamically changing usage pattern is a much less studied technique. The idea is suggested in principle by Lin and Zheng [7]. However, no detailed explanations and precise formulations are provided on how to execute the approach and implement the solution. Our work is a first attempt to combine the JPEG 2000 flexibility with a user's behavior model to jointly manage a cache and prefetch data.

The remainder of the paper is organized as follows. Section 2 describes the modeling assumptions. Section 3 presents the simulations experiments for three different network configurations and highlights the improvement provided by the proposed approach. Section 4 concludes the paper and exposes further research directions.

2. MODELING ASSUMPTIONS

2.1. Key elements from JPEG 2000

An image encoded with the JPEG 2000 algorithm can be seen as a succession of packets. Each of these packets contains a certain amount of data corresponding to one and only one spatial zone (called *precinct*), resolution and quality level of the image. Moreover, a hierarchy exists in these packets: to display a precinct i at resolution b and quality n, you also need the packets of the

[a] A.Descampe is funded by the Belgian NSF (FNRS).

The picture used for the experiments has been provided by SPOT Image in the scope of IST-2000-28646 PRIAM project.

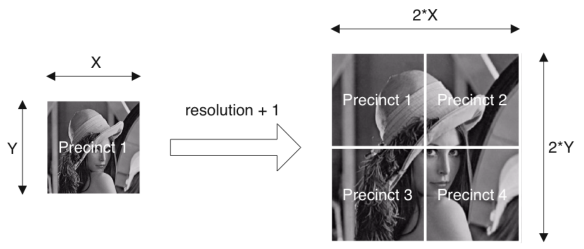
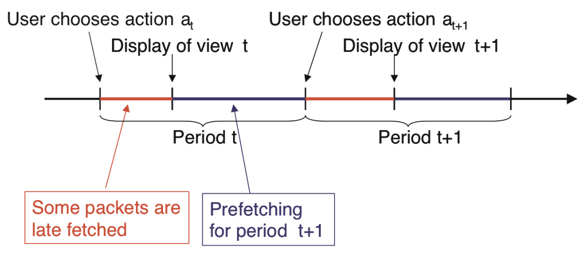


Fig. 1. The precinct size is fixed so that precincts cover smaller spatial zones as the resolution increases.



User chooses action a_t
Display of view t
User chooses action a_{t+1}
Display of view t+1
Period t
Period t+1
Some packets are late fetched
Prefetching for period t+1

Fig. 2. Timing of events.

precincts that contain i for all resolutions $a < b$ and all quality levels $m < n$.

Whatever the resolution level, we fixed the precinct size to the size of the Window Of Interest (WOI), which is a constant. This means that if the entire image at the lowest resolution level contains one precinct, the upper one will contain four and so forth (Fig. 1).

2.2. Model notations

We consider the succession of JPEG 2000 packets as a set of data blocks, indexed as $\mathcal{D} = \{d_k, k = 1, \cdots, K\}$, with each block representing a particular precinct at certain resolution and quality level. The sizes of the data blocks can vary greatly, but are known a priori and denoted as $s_k, k = 1, \cdots, K$. At time 0 the viewer is presented with a default coarse view of the entire image, which comprises a subset $D_0 \subset \mathcal{D}$ of data blocks that are always initially transmitted. At time $t > 0$ the viewer takes an action a_t and waits for the corresponding set of blocks $D_t \subset \mathcal{D}$ to be fetched and displayed. Once the view is displayed the user will watch it for some time. During that time the system will prefetch additional blocks. Timing of these events is illustrated in Fig. 2. As it can be seen, navigation will be smoother if we minimize the amount of data that is late fetched.

To model the costs associated with the wait of the viewer when non-prefetched blocks have to be transmitted on the spot and the bandwidth waste when some prefetched blocks are never used, we assign a linear cost coefficient w to represent the viewer waiting cost and a linear cost coefficient c to represent the transmission cost. Both are given per unit of data. So the cost to transmit data block d_k is $c_k = cs_k$, and the waiting cost incurred by the viewer for transmitting data block d_k on the spot is $w_k = ws_k$.

Finally, we define b as the size of the cache that can hold prefetched data, $B_t \subseteq \mathcal{D}$ as the subset of data blocks that are in the cache at time t, r as the transmission rate between server and client, and l_t the time the user watches view t. In general l_t can be a random variable, however, in this paper we assume it is a known constant. The amount of data that can be prefetched during period t is given by $r_t = rl_t$.

2.3. User's behavior

In the present study, we assume that the quality level is fixed during the navigation. Consequently, the user can only browse through various resolutions (zoom-in, zoom-out actions) and various spatial locations (up, down, right, left actions): s/he can take 6 possible actions to choose next view. We assume that user's actions can be approximated by a 1-degree autocorrelation model. Specifically:

- the same action will be chosen with probability α
- all other actions have probability $\frac{(1-\alpha)}{5}$

Based on this model the server can calculate, in theory for every future time $s > t$, the probabilities $p_{kts} = P[d_k \in D_s | D_t, a_t]$ that data block d_k will appear in display D_s at time $s > t$ given the present display D_t and the last move a_t. However, for our practical implementation, we decided to deploy only solutions for a "myopic" one-step look ahead problem so only $p_{kt,t+1}$ is actually calculated, which can be done on the fly on modern server computers.

2.4. Decision variables

At time t the decisions to be made in the modeling framework are summarized by the following two groups of variables:

1. $x_{kt}, k = 1, \cdots, K$, and $d_k \notin B_t$: $x_{kt} = 1$ if we decide to prefetch data block d_k at time t, $x_{kt} = 0$ otherwise;
2. $z_{kt}, k = 1, \cdots, K$, and $d_k \in B_t$: $z_{kt} = 1$ if data block d_k is not erased, $z_{kt} = 0$ otherwise.

These will be the decision variables in the optimization problems presented in the next section, together with the simulation results.

3. SIMULATION EXPERIMENTS

Now that the modeling framework has been set up, we can apply it to three client-server systems of distinctive characteristics, presented in Table 1.

As a benchmark to check how much our prefetching strategy can help we will consider two more basic policies,

1. the first one is no prefetching: we do only caching of previously displayed precincts. If needed, we remove the least recently used (LRU) data blocks. This is the standard cache management rule [8].
2. the second rule is based on the isotropic behavior of the user only. In this case, we do not know the user's behavior and always suppose that all the possible moves have the same probability when computing the probabilities of the data blocks.

The image used for the simulations is a $8k \times 8k$ digital map of the center of Toulouse (France) and the WOI was set to 256×256. The compressed file size is about 42 MB and the largest amount of data to transmit one precinct is about 270 KB. We ran each simulation for 5000 moves.

	Typical platform	Cache size	Trans. rate	Trans. cost
System 1	Wireless device with high speed connection (blue tooth)	$b < \infty$	$r = \infty$	$c = 0$
System 2	Personal computer with limited speed connection	$b = \infty$	$r < \infty$	$c > 0$
System 3	Wireless device with slow connection (mobile phone)	$b < \infty$	$r < \infty$	$c > 0$

Table 1. Characteristics of the three client-server models that have been tested.

3.1. System 1

The first system is the simplest, and provides a basic model that extends through generalization on different dimensions to the other two systems. We want to remark that the zero transmission cost assumption in this system is consistent with the assumption of an unlimited (or extremely large) transmission bandwidth. This allows us to assume that the cache can be refreshed in every prefetch period. Thus the optimization problem for System 1 involves only the first group of decision variables:

$$\begin{aligned} \text{Min} \quad & \sum_k w_k p_{kt,t+1}(1 - x_{kt}) \\ s.t. \quad & \sum_k s_k x_{kt} \leq b \end{aligned} \tag{1}$$

where the objective function is to minimize the expected waiting cost. The problem is equivalent to

$$\begin{aligned} \text{Max} \quad & \sum_k w_k p_{kt,t+1} x_{kt} \\ s.t. \quad & \sum_k s_k x_{kt} \leq b \end{aligned} \tag{2}$$

This is a standard knapsack problem. Considering the large size b of the cache compared to the sizes of individual data blocks, we can easily approximate the optimal solution by selecting data blocks for prefetching in decreasing order of the ratio

$$\frac{w_k p_{kt,t+1}}{s_k} = w p_{kt,t+1}, \tag{3}$$

subject to the cache size constraint.

Fig 3 shows the simulation results for System 1. As we can see, if the cache is large enough (large enough to contain all the data needed for the largest precinct) then prefetching will really help. Simulations were indeed also ran for cache sizes smaller than 180 KB but then there were hardly any difference with the LRU rule. Moreover, the more predictable the user, the larger the benefit from anticipating the user's behavior is.

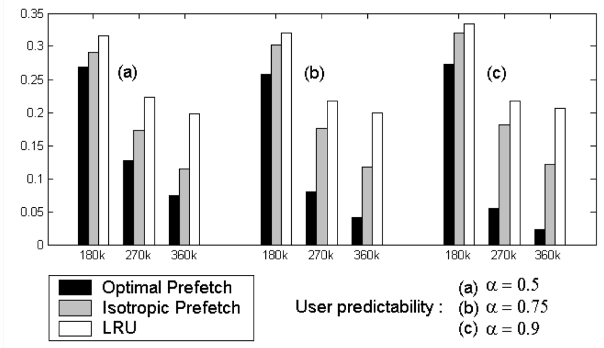


Fig. 3. Results for System 1. The Y-axis give the fraction of packets to be retrieved after prefetch. The X-axis corresponds to various combinations of cache size and user predictability.

3.2. System 2

Now we extend the basic model (1) to systems with limited transmission bandwidth between the server and the client and positive transmission costs. In this second system we add in the assumption that the cache size is infinite (or big enough to hold the entire image) so we never erase anything from the cache. Restricting again our policy to a myopic one period look-ahead problem, it can again be formulated as a knapsack problem:

$$\begin{aligned} \text{Min} \quad & \sum_{k \notin B_t} (c_k + w_k) p_{kt,t+1} \\ & + \left(\left(1 - p_{kt,t+1}\right) c_k - w_k p_{kt,t+1} \right) x_{kt} \\ s.t. \quad & \sum_{k \notin B_t} s_k x_{kt} \leq r_t \end{aligned} \tag{4}$$

The optimal solution can again be approximated by selecting data blocks for prefetching in decreasing order of the ratio

$$\frac{w_k p_{kt,t+1} - \left(1 - p_{kt,t+1}\right) c_k}{s_k}, \tag{5}$$

subject to the transmission bandwidth constraint. Notice that the ratio of (5) extends that of (3) by taking into account the possible bandwidth waste cost if block d_k is not used in the display D_{t+1}.

In a first experiment, we consider only a limitation on bandwidth ($c_k = 0$) so that the system is similar to the first one except that the main constraint is the bandwidth and not the cache.

Fig 4 shows the simulation results for System 2. As we can see, even with a small bandwidth, the prefetching will significantly reduce the user wait. On the other hand, the predictability of the user makes much less difference. If the user has larger probability of repeating a move, the amount of "fresh" data will be larger and the limited bandwidth does not make it possible to prefetch all the data even if we know we will need it.

In our second experiment for System 2, we study what happens when the bandwidth is unlimited and the transmission cost positive. We can formulate the problem as a newsboy problem: we transmit only blocks with a probability of being displayed $p_{kt,t+1} > \frac{c}{(w+c)}$. With $c = w$, we should obtain a minimal cost if we

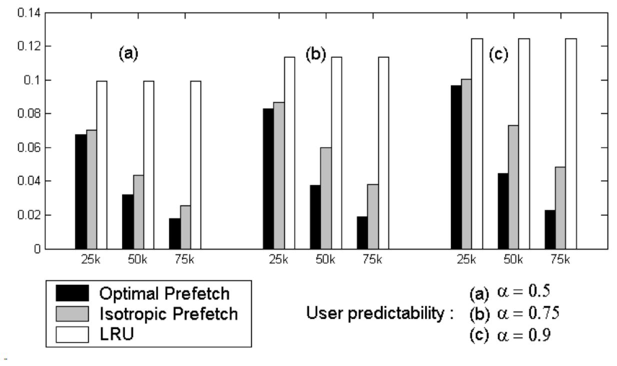


Fig. 4. Results for System 2. The Y-axis give the fraction of packets to be retrieved after prefetch. The X-axis corresponds to various combinations of bandwidth and user predictability.

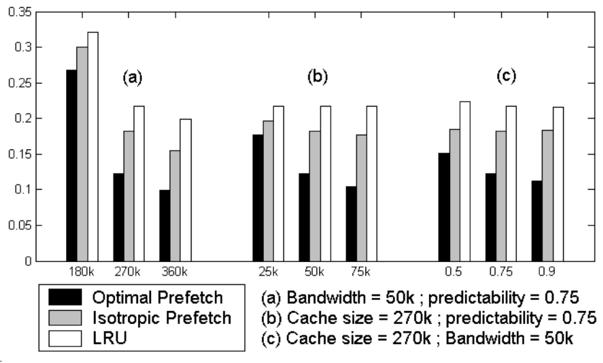


Fig. 5. Results for System 3. The Y-axis give the fraction of packets to be retrieved after prefetch. The X-axis corresponds to various combinations of bandwidth, cache size and user predictability.

prefetch only data blocks d_k if $p_{kt,t+1} > 0.5$. But as there are very few blocks with a probability close to 0.5 this is hardly noticeable in the results obtained (not shown here). To more uniformly spread the blocks probabilities over the $[0;1]$ interval, we would need a finer prediction of the user's behavior.

3.3. System 3

Finally we extend the basic model (1) to systems with limited cache size and limited transmission bandwidth. We must now include the possibility that some data blocks are removed from the cache to make room for other blocks. As such, the optimization problem involves the two groups of decision variables:

$$\begin{aligned} \text{Min} \quad & \sum_{k \notin B_t} (c_k + w_k) p_{kt,t+1} \\ & + \left((1 - p_{kt,t+1})\, c_k - w_k p_{kt,t+1} \right) x_{kt} \\ & + \sum_{j \in B_t} (c_j + w_j) p_{jt,t+1} (1 - z_{jt}) \\ s.t. \quad & \sum_{k \notin B_t} s_k x_{kt} + \sum_{j \in B_t} s_j z_{jt} \leq b \\ & \sum_{k \notin B_t} s_k x_{kt} \leq r_t \end{aligned} \qquad (6)$$

In this case we can adapt the policies from the previous cases:

- We prefetch the data blocks most likely to be displayed until we reach the bandwidth limit or the next data block has a display probability smaller than the lowest probability block in cache.
- When the cache is full we remove the blocks least likely to be displayed next period.

As it can be seen in Fig 5, as long as the cache is large enough and there is sufficient bandwidth the prefetching policy improves the navigation significantly.

4. CONCLUSION AND FURTHER RESEARCH

Taking advantage of the JPEG 2000 encoding structure can make navigation in large images significantly smoother. As we can see in the simulations results, even relatively simple myopic policies make a significant difference. Further research should investigate whether a policy that anticipates the user's behavior over more than one period improves the system even more. It could also include the quality layer paradigm which was not used in this model. For example, quality levels could be determined according to an "interest ratio" computed by the system. This would need of course to include mechanisms to know and retrieve data with a potential interest for the user.

5. REFERENCES

[1] R. Ferreira *et al.*, "The virtual microscope," in American Medical Informatics Association, 1997 Annual Fall Symposium, pp 449-453, Nashville, TN, 1997.

[2] Eugenia A. Politou, George P. Pavlidis, and Christodoulos Chamzas, "JPEG2000 and Dissemination of Cultural Heritage Over the Internet," IEEE Trans. on Image Processing, vol.13, no.3, pp 293-301, March 2004.

[3] ISO/IEC 15444-1: Information Technology-JPEG 2000 image coding system-Part 1: Core coding system, 2000.

[4] J. Wang *et al.*, "Multiresolution browsing of pathology images using wavelets," Proc. of AMIA Symposium, pp 430-434, 1999.

[5] R. Prandolini, "15444-9:2004 JPEG 2000 image coding system - Part 9: Interactivity tools, APIs and protocols - (JPIP) Common Text," Tech. Rep. N3358-r2, ISO/IEC JTC1/SC29/WG1, July 2004.

[6] A. Ortega *et al.*, "Soft caching: Web cache management for images," IEEE Signal Processing Society Workshop on Multimedia Signal Processing, Princeton, NJ, pp 23-25, June 1997.

[7] C. Lin and Y. F. Zheng, "Fast browsing of large scale images using server prefetching and client caching techniques," Proc. of SPIE, Applications of Digital Image Processing XXII, pp 376-387, Denver, Colorado, July 1999.

[8] J. Robinson and M. Devrakonda, "Data cache management using frequency-based replacement," Proc. ACM SIGMETRICS '90, pp 134-142, 1990.