L/MRP: A Buffer Management Strategy for Interactive Continuous Data Flows in a Multimedia DBMS

Frank Moser, Achim Kraiß†, Wolfgang Klas

GMD – Integrated Publication and Information Systems Institute (IPSI)
Dolivostraße 15, D–64293 Darmstadt, Germany
{moser, klas}@darmstadt.gmd.de

Abstract

Multimedia applications demand specific support from database management systems due to the characteristics of multimedia data and their interactive usage. This includes integrated support for high–volume and time–dependent (continuous) data types like audio and video. One critical issue is to provide handling of continuous data streams including buffer management as needed for multimedia presentations. Buffer management strategies for continuous data have to consider specific requirements like providing for continuity of presentations, for immediate continuation of presentations after frequent user interactions by appropriate buffer resource consumption. Existing buffer management strategies do not sufficiently support the handling of continuous data streams in highly interactive multimedia presentations. In this paper we present the "least/most relevant for presentation" (L/MRP) buffer management strategy which considers presentation specific information in order to provide an optimized behavior with respect to the requirements mentioned above. L/MRP is a framework to formulate specific interaction models and is therefore adaptable to individual multimedia applications. We present a simulation study showing that an instantiated L/MRP outperforms existing approaches for given types of interactive multimedia applications. It is shown that L/MRP is especially suitable to support highly interactive multimedia presentations.

1. Introduction

Database systems can serve as a suitable platform for a wide spectrum of multimedia applications, which includes News–on–Demand, Video–on–Demand, Multimedia Systems Engineering, Media Editing Workbenches, Computer Supported Instruction & Training Systems, Technical Documentations, Animations, and many more. Multimedia data processed by such applications are often embedded into other conventional types of data, e.g., in form of hypermedia documents. This often requires a database system to provide *integrated* handling of multimedia data. Appropriate system services with respect to e.g., data volume and time-dependency of continuous data have to be realized. Another crucial issue is the fact that many of the applications mentioned are characterized by a potentially high degree of user interactions. Examples for such interactions are *rewind* and *fastForward* of a video presentation, *jump* to a specific video frame, etc.. For instance, in a video editing workbench, an editor jumps to marked positions, rewinds to certain scenes, or skips uninteresting information by using appropriate interactions like jump, rewind and fastForward, respectively. Another example is a computer supported instruction/training system which may present very complex structured information too fast to a student causing him to repeat the presentation of specific parts by invoking interactions like pause, fastBackward, and (normal) playForward, etc..

Interaction response time, i.e., the delay between the occurrence of an user interaction and the time when the system reacts on this interaction by continuing with the presentation flow, is a critical parameter for the usage and acceptance of multimedia systems. Thus, a multimedia database system has to support interactive multimedia applications by keeping the interaction response time at a minimum.

In order to be able to support directly interactive and time–dependent data, a multimedia database system internally has to provide a continuous flow of so–called *Continuous Object Presentation Units* (COPU). During a presentation COPUs have to be loaded into buffer before they will be accessed from higher–level modules of the database system. Examples for such modules are a presentation or a synchronization manager. In the case of a client/server architecture the COPUs in the server buffer are consumed by a network interface

† The new address of the author is Department of Computer Science, University of the Saarland, D–66041 Saarbrücken, Germany. kraiss@cs.uni–sb.de.

Proceedings of the 21th VLDB Conference
Zurich, Switzerland, 1995

in order to deliver them in time to the presenting client [22]. We call the (lower–level) module responsible for the management of continuous data the *Continuous Object Manager* (COM). It has to provide object management functionality including storage and retrieval of continuous objects, realization of data placement strategies on storage devices, and management of adaptive, continuous data flows. We call the higher–level modules requesting and accessing COPUs by using the interface of the COM *consuming* modules.

Two main issues of object buffer management are *preloading* and *replacement* of continuous data. Preloading is necessary in the case of non–real–time behavior of the underlying system components (e.g. storage devices or networks). COPUs needed by a consuming module are to be kept in buffer *before* they are requested. A loading on demand strategy can not guarantee continuity and would lead to a jittery presentation. The number of COPUs to preload depends on the predicted loading time of the external disks or network connections and determines the initial delay of a presentation. Strategies for quantifying this parameter can be found e.g. in [9], [17].

The main goal of a replacement strategy is to replace those COPUs from buffer which are expected to be unreferenced, i.e. not to be presented, for the *longest* period of time in the future. Assuming a single, non–interactive presentation of the continuous object, the strategy of tossing a COPU immediately after it was presented is optimal. By taking into account the *interactivity* of multimedia presentations the replacement strategy has to consider the effect of user interactions on the data flow. Before the presentation can continue, the COM has to preload COPUs in order to guarantee continuity, also for the next presentation. Thus, interaction response time is primarily determined by the number of buffer faults occurring during the initial preloading phase. In order to reduce the number of buffer faults the buffer management strategy has to consider potential interactions by keeping those COPUs in buffer which are potentially referenced after likely interactions. As a matter of fact previously presented COPUs may become referenced after an interaction.

Note that supporting interactions requires additional buffer space. Besides the preloaded COPUs needed for continuity, the buffer must also keep those COPUs, which are referenced with high probability after interactions. It should be possible to tune the buffer management strategy in its degree of interaction support to minimize buffer consumption. In the extreme of none interaction support it degenerates to a simple "Use&Toss" [4].

In this paper we propose the integrated preloading and replacement strategy L/MRP (*least/most relevant for presentation*) for a COM of a multimedia database system suitable for the management of both highly and less interactive continuous data flows needed in multimedia presentations. The L/MRP strategy is adaptable to specific application semantics characterized by different interaction patterns and probabilities.

The paper is organized as follows. In section 2 we give a motivating example for the behavior of a COM handling an interactive multimedia presentation and discuss related work. In section 3 the general L/MRP loading and replacement algorithm will be described. In section 4 we give preliminary performance results of L/MRP in comparison to conventional, state–of–the–art strategies. Section 5 concludes the paper and gives an outlook to future work.

2. Motivation and Related Work

Let us assume the following example application in order to illustrate the usage of interactions and the resulting data flows. In a Multimedia Database we have stored (Motion–JPEG–) videos about sightseeing–tours of cities. In an specific application program the user can query the database and start the presentation of a selected video. For interactive presentation the application provides a VCR–like user interface with buttons including *playForward*, *playBackward*, *fastForward*, *SetWP RemoveWP* and *JumpWP*. The buttons realize a normal play, a backward play, a fast forward play, the setting and removing of working points and a direct jump to a previously selected working point, respectively. We assume that working points can be named and that they are given by the current position within the video at the time the *SetWP* button is pushed.

User interactions determine the progress of the presentation. Each interaction is signalled to the COM of the multimedia database system which reacts on the interaction by starting a corresponding data flow. Figure 1 illustrates an example state of a data flow with one working point *wp*.

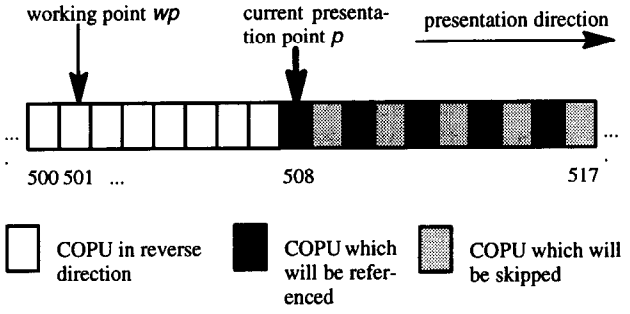

Figure 1: Example state of a data flow

All COPUs of a continuous object are indexed from 0 to $n-1$ with n denoting the total number of COPUs. In our example, the working point *wp* was set at index 501. The current presentation point is at 508 and it moves towards the end of the object, i.e., the current presentation direction is forward. We denote the direction and skip parameter of a presentation by a single signed *skip value*, positive for forward, negative for backward direction and its absolute value denoting the skip. Our example presentation can be character-

ized with a skip value of +2 i.e., the current direction is forward and only every second COPU is presented (which is assumed to realize fastForward). We can easily identify three different COPU types: COPUs being located in the reverse direction, those being referenced in future and those being skipped because of a skip value greater than one. If the user requests a playBackward operation, the COM has to deliver a data flow which starts at position 507 and which has a skip value of −1 because direction reverses and speed has to be normal. If the user pushes the playForward button this will lead to another data flow starting with COPU 509 with a skip value of +1 because direction is the same as before but speed has to change to normal. If the user decides to jump to the working point *wp* the COM simply has to load COPU 501 and waits for a further interaction, e.g., playForward.

We assumed above that we have an abstract access granularity called COPU. At the level of the COM a continuous object consists of a sequence of COPUs, which are not further interpretable at this level, i.e., they are simple bulks of bytes. This low level of abstraction is sufficient to realize all the continuous object management functionality, and we argue that this concept is suitable for most multimedia data types. For example, a Motion–JPEG sequence is modelled as a sequence of COPUs each of which representing a JPEG–compressed frame of variable size. A PCM–encoded audio can consist of COPUs each of which representing a block of PCM–compressed samples of fixed size. A subtitle sequence of a video can be seen as a sequence of COPUs each of which representing an ASCII–formatted string of variable size. MPEG–Video [13] sequences could be modelled by defining a group of pictures as a variable–sized COPU. But this would lead to some problems in realizing those interaction primitives as explained above. The approach of modelling the various MPEG picture types (I–, P– and B–frames) in different sequences of COPUs seems to be more flexible. In this case, an abstract data type for MPEG-videos has to be defined in order to hide the separation of the three sub–data flows. As a COM only deals with COPUs and not with different, concrete compression or data formats it is flexible enough to be used for various multimedia data types.

Many multimedia systems use the "Use&Toss" replacement strategy as proposed in [4]. In this strategy each COPU is free for replacement immediately after it was presented. The drawback of this simple strategy is that COPUs which are potentially referenced after an user interaction are not kept in buffer. For example, if the user initiates a playBackward interaction from the presentation state playForward, all previously tossed COPUs may have to be reloaded into buffer, which leads to increased interaction response times. If no interactions occur, the "use&toss" strategy is optimal due to the linear reference pattern.

Another related field are replacement strategies considering multiple concurrent presentations. In [14] and [23] buffer management strategies are proposed suitable for media sharing among multiple viewers. The replacement strategies consider presentations following the leading one and thus keep COPUs in buffer needed by presentations that will reference them in near future. Interactions are only considered in the sense that they change the timely distances between viewers and thus the resulting data flows are limited to fixed intervals. In this paper we focus on a single interactive continuous data flow.

There has been a lot of work in the field of replacement strategies for conventional database applications. They can be differentiated in (a) strategies using general heuristics or (b) strategies using semantic information about the running application.

There are many papers analyzing and comparing heuristic replacement strategies like LRU, Fifo, LFU, etc., ([7], [6], [18]) and further improvements and refinements of them ([19], [12]). It is obvious that all these strategies do not explicitly address the reference behavior of interactive continuous data flows. Furthermore, presentation scenarios can be constructed where the heuristics show destructive behavior, i.e. systematically remove COPUs from buffer needed in near future. We want to show this by a little example. Suppose our buffer uses the LRU strategy. We begin with an empty buffer with constant buffer size 15. Playing frames 1 to 20 leads to a buffer state in which frames 6 to 20 are in buffer after the presentation has finished. Let us consider as the next presentation a playForward from frame 5 to 15. Frame 5 is not present in buffer which causes a buffer fault. The least recently used frame is 6. LRU replaces 6 (!) to load 5. Now we request frame 6, and LRU replaces 7(!), and so on. LRU always replaces the frame that will be needed next. The reason for this behavior is that the general purpose *heuristic* LRU does not consider (and cannot) consider any presentation specific information. However, we do not claim that in this specific example none of the "classical" strategies fits well (actually, the Most Recently Used strategy works well), but for all these heuristic-based strategies similar examples of "misbehavior" can be found. Although it seems that the destructive behavior is of significance for some "constructed" examples only, we show in our performance investigations given in chapter 4 that for a Video on Demand scenario RANDOM outperforms LRU indicating the average misbehavior of LRU in our application.

Unlike heuristics–based strategies, the Hot–Set [24] and the DBMIN [3] strategy take into account access patterns, whereas the Marginal Gains approach [16] considers also the run–time availability of buffer resources. Hint–passing algorithms consider some system specific information like indexes [1], [11]. The L/MRP strategy we propose in this paper can be seen as another application–specific strategy, but especially designed and suitable for interactive continuous data flows.

In the next section we describe the L/MRP buffer management strategy for interactive continuous data flows in detail.

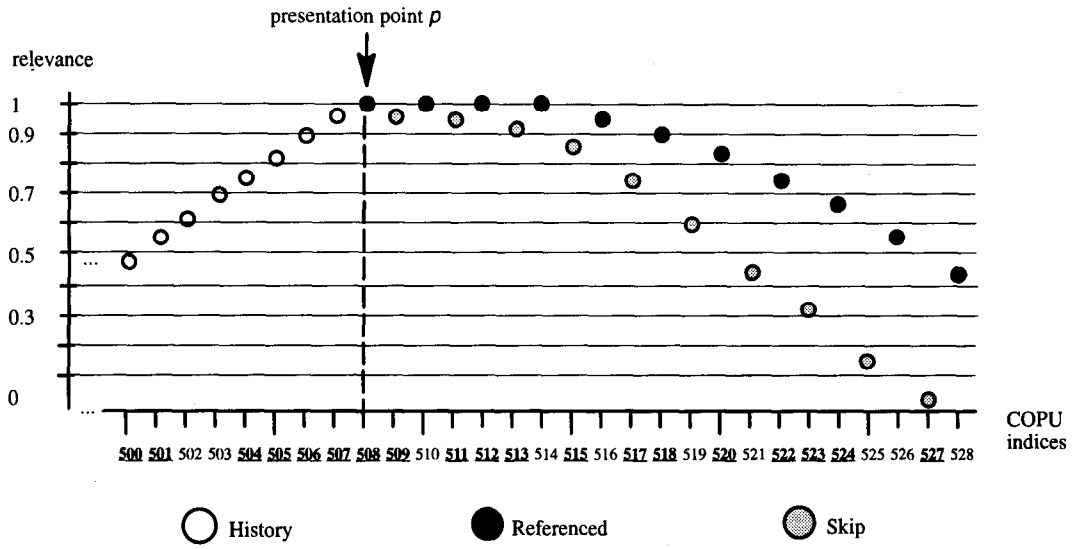

Figure 2: An example of interaction sets with relevance values

3. Replacement and Preloading Strategies for Continuous Objects

By considering presentation information on the level of buffer management, we are able to optimize the replacement and preloading strategies for interactive data flows. Our strategy considers so called *interaction sets*. Some examples of those interaction sets were already depicted in Figure 1. These are the sets of future referenced COPUs (*Referenced*), COPUs in reverse direction (*History*) and COPUs which will be skipped (*Skip*). By introducing the notion of relevance function, we will give every element of a set a value to denote its significance for replacement and preloading. A replacement and preloading strategy can be defined, which makes use of those relevance values in the way that least relevant COPUs are replaced and most relevant COPUs will be preloaded. The relevance value of a COPU depends also on specific presentation parameters like the number of the currently presented COPU. Consequently, we call our buffer management strategy *least/most relevant for presentation (L/MRP)*.

In order to explain the general idea of L/MRP, we use the same presentation snapshot as already illustrated in figure 1. Figure 2 shows the interaction sets *History, Referenced* and *Skip* for the given presentation point (508) and the given skip value specific for the example (+2). Each COPU, identified by the index on the x–axis, is associated with a relevance value reflecting the importance of the COPU with respect to the specific interaction set. Note that a COPU of an interaction set can become referenced by the occurrence of no interaction (*Referenced*) or one interaction (*Referenced, Skip* and *History*). The settings of the relevance values should reflect the access probability of a COPU for this cases.

Let us assume that due to some previously ongoing presentations, the COPUs with bold, underlined numbers are in buffer. The least relevant COPUs, i.e. the COPUs with lowest relevance value, at this moment are 527, 525, 523, 528, 521, 500, 526, etc.. Thus, the next replacement candidates are 527, 523, 500 , etc.. The most relevant COPUs for presentation are 508, 510, 512, 514, etc.. Thus, the next preloaded COPUs will be 510, 514, etc.. In the following we will give some formal definitions which are useful to introduce L/MRP.

A General Model for L/MRP

Let CO (continuous object) denote the sequence of all COPUs which constitute a continuous object. An element c_i, $i = 0,...,|CO|-1$, denotes the COPU with index i within the continuous object CO. The state of a presentation is characterized by a tuple $s = \langle p, skip \rangle$, with $p \in N_0$ denoting the index of the COPU at the current presentation point and $skip \in Z$, $skip \neq 0$, denoting the skip value. Later we will extend the definition of s by considering further presentation information like the positions of working points. COPUs are related to one or more interaction sets $A_{k,s}$, $k = 1, 2, ..., n$. Every set $A_{k,s}$ has associated a criterion used to decide, whether a COPU belongs to the set at a specific point of a presentation or not, i.e., the membership of a COPU to a set depends on the presentation state s. We want to express in $A_{k,s}$ not only the membership of a COPU with respect to s, but also its relevance. Hence, an interaction set $A_{k,s}$ is defined as a binary relation relating a COPU to a relevance value. For example the interaction set $Referenced_s$ with s=<508,+2> as visualized in Figure 2 is:

$Referenced_{<508,+2>} = \{(COPU_{508},1),(COPU_{510},1),(COPU_{512},1),(COPU_{514},1),(COPU_{516},0.95),...\}$.

To denote the relevance of a COPU within a given interaction set $A_{k,s}$, we define a *distance relevance function* dr_A. The domain of a distance relevance function are relative

positions of COPUs independent to p, the current presentation point. Functions dr_A map distances to values in]0,1]:

dr_A: $N_0 \mapsto]0,1]$.
$dr_A(i)$ is the relevance value of a COPU with distance i to any possible presentation point p.

For example the distance relevance function for $dr_{History}$ in Figure 2 is a monotonous decreasing, linear function.

The distance relevance values describe the degree of importance to keep COPUs in buffer. For example, the distance function $dr_{Referenced}$ for all future referenced COPUs describes the degree of importance to keep specific COPUs in buffer because of the high probability to be accessed in the current presentation. A distance relevance function value of 1 means that the COPU is most relevant for presentation, and, hence, is not to be considered as candidate for replacement, but has to be preloaded by L/MRP, if necessary. The motivation to use distances instead of COPU indices lies in its flexibility, because distances are presentation state independent. Therefore, general application specific semantics can be captured in an easier way.

The definition of concrete distance relevance functions is typically the task of an application database administrator. The tuning of the L/MRP strategy to different interactive applications can be realized by specifying different distance relevance functions. Furthermore, one can imagine that the determination of application specific distance relevance functions can be trained by help of learning tools.

Let us now give the definition of the interaction set A_s for a presentation state s:

$$A_s = \{ (c_j, dr_A(i)) \mid c_j \in CO,\ j = g(i,s),\ i \in N_0 \}.$$

The index j of a COPU to be considered in A_s is determined by a function g which depends on the distance of the COPUs i and the current presentation state s. The relevance value for a COPU c_j is determined by the distance relevance function dr_A. We will show later how the interaction sets $Referenced_s$, $Skip_s$, $History_s$ and $Point_s$ can be defined accordingly.

To compare the relevances of COPUS with respect to the whole continuous object we introduce the *relevance function* r_{As} for an interaction set A_s:

$$r_{A_s} : CO \mapsto [0, 1].$$

$$r_{A_s}(c) = \begin{cases} v\ , & (c, v) \in A_s \\ 0\ , & \text{otherwise} \end{cases}$$

The relevance function can be obtained by projection on the second position of the respective interaction set, if a COPU is considered there; otherwise a value of zero is assigned.

COPUs can be considered in multiple specific interaction sets. In order to determine the overall relevance of a COPU with respect to all participating interaction sets $A_{k,s}$, $k = 1, 2, \ldots, n$, we introduce the function r_{CO} which computes for each COPU of the continuous object its overall relevance value. With the help of r_{CO} the replacement and preloading victims are determined.

$$r_{CO,\,s} : CO \mapsto [0, 1].$$

$$r_{CO,\,s}(c) = \max_{k = A_{1,s}, \ldots, A_{n,s}} (r_k(c))$$

The function r_{CO} specified here describes the maximum closure of the relevance values of all COPUs. Alternative definitions of r_{CO} are imaginable. For example one could use an operator which calculates the minimum of 1 and the total sum of all the relevance values $r_k(c)$, $k = A_{1,s}, \ldots, A_{n,s}$ instead of the plain maximum operator.

The identification of interaction sets for continuous objects and the definition of distance relevance functions are strongly dependent on the application scenario. For example, for a video editing workbench it can be assumed that specific editing operations are performed on any part of the video, whereas for a Video on Demand service the system only has to support a few default operations like *playForward* or *fastForward*. A detailed discussion of typical application scenarios with respect to interaction sets which are to be considered and of feasible relevance functions is not within the scope of this paper, but we will motivate why our approach can be adapted to different application needs.

In the following we give the definition of the interaction sets $Referenced_s$, $Skip_s$, and $History_s$. In addition we show how one can define a generic interaction set to consider any number of selected working points ($Point_s$). We think that these generic interaction sets are useful for quite a lot of multimedia application scenarios. Thus, they describe the skeleton of a COM for general purpose use.

(a) In Future Referenced COPUs

During a presentation, there might be already COPUs in buffer which are going to be referenced, if the presentation status does not change due to a user interaction. COPUs located nearer to the actual presentation point have a higher probability to be referenced.

The COPUs of the interaction set *Referenced* can be obtained directly from the presentation state s. The relevance function values are determined by the distance relevance function $dr_{Referenced}$.

$$Referenced_s = \{ (c_j, dr_{Referenced}(i)) \mid j = p + i \cdot skip,\ c_j \in CO,\ i \in N_0 \}.$$

According to this definition the function g (introduced in the general definition of A_s) which determines the index j depends on i and the presentation state s, i.e., the current presentation point p and the skip value $skip$. An extended definitions of s can be desirable, e.g., by considering in s COPUs belonging to a commercial. If one is only interested in pres-

enting COPUs within a video data stream which are not part of a commercial, the definition of the interaction set $Referenced_S$ can be adapted by ignoring those COPUs with the effect that these COPUs get a relevance value of zero.

A reasonable choice of the distance relevance function $dr_{Referenced}$ is a monotonous decreasing function. Due to the independence of the distance relevance function from presentation parameter the relevance values are correspondingly decreasing along the presentation direction. The motivation for this is that COPUs with higher relevance values are kept in buffer, and referenced COPUs located nearer to p should not be replaced. This is a direct consequence coming from the OPTIMAL [2] replacement strategy, which says that the COPU with the longest future reference distance should be replaced. Other appearances of the distance relevance function can be appropriate, especially, if other application specific knowledge is available, e.g., if users tend to skip the neighborhood region from the actual presentation point with a high probability (user–controlled scanning). In this case a respective minimum of the distance relevance functions can reflect this behavior.

To guarantee continuity of presentations, we assign COPUs considered in *Referenced* and located "near" to the current presentation point a relevance value of 1. This means that those COPUs are most relevant for presentation and, therefore, are not subject for replacement, but on the contrary have to be considered for preloading, if necessary. This implicitly defines a channel which contains prefetched COPUs, which will be delivered by the COM to the consuming component. In the following, let f denote the number of COPUs the COM has to prefetch in order to guarantee continuity. An example for a quantification of f can be found in [9]. Obviously, f must be restricted depending on the size of CO and the *Buffer_Size* which denotes the maximum number of COPUS which can be stored in buffer. Taking f into account a database application designer could define $dr_{Referenced}$ as follows:

$$dr_{Referenced}(i) =$$

$$\begin{cases} 1, & 0 \leqq i \leqq f < \min(|CO|, Buffer\_Size) \\ 1 - \alpha \cdot i, & f + 1 \leqq i < \min(|CO|, Buffer\_Size), \\ & 0 < \alpha < \dfrac{1}{|CO| - 1}. \end{cases}$$

The function $r_{Referenced_S}$ follows the general definition of the relevance function r_{A_S} introduced earlier.

(b) Potentially Referenced COPUs by Skip Change

For skip values greater than 1, there might be COPUs which will be referenced in the case of an interaction affecting the skip value. Consider a *fastForward* realized with a skip value of 2. A presentation change to *playForward* imposes that neighborhood COPUs are needed immediately. The set *Skip* contains COPUs which can be referenced in the future, if the skip value will change. Even in the case of presenting an audio a skipping strategy might be appropriate. A fastplay of an audio can be realized by successively playing part of the audio pieces. A COPU would represent such an audio piece; the skipped COPUs between two played audio pieces are the part of the audio that is ignored. The set *Skip* can be defined as follows.

$$Skip_S = \{ (c_j, dr_{Skip}(i)) \mid c_j \in CO, |skip| > 1, i \in N_0,$$
$$j = p + \frac{skip}{|skip|} \cdot (1 + i + \left\lfloor \frac{i}{|skip| - 1} \right\rfloor).$$

The set $Skip_S$ contains all COPUs in presentation direction which are not considered by $Referenced_S$.

The function dr_{Skip} could be a monotonous decreasing function for the same reason as motivated in (a). The following condition has to hold: $0 < dr_{Skip}(i)$. Further constraints may be imposed, e.g., $dr_{Skip}(i) < dr_{Referenced}(i)$, for all $i \in N_0$. The function r_{Skip_S} follows the general definition of r_{A_S} introduced earlier.

(c) Potentially Referenced COPUs by Direction Change

Suppose a user changes the presentation direction. The actual presented COPU c_p is considered in the interaction set $Referenced_S$. The COPU c_j which preceded c_p ($j < p$ or $j > p$ depending on the original presentation direction) was considered in the interaction set $Referenced_S$, but after its presentation it has not belonged to this or to any other interaction set. The relevance of c_j for keeping it in buffer would be zero. However, in order to support the change of the presentation direction in the sense to prevent preloading of already displayed COPUs we define an interaction set $History_S$ which contains all COPUs which can be referenced after a change of direction. The set $History_S$ can be defined as follows:

$$History_S = \{ (c_j, dr_{History}(i)) \mid c_j \in CO, i \in N_0, \quad j = p - (i + 1) \cdot skip / |skip| \}.$$

The function $dr_{History}$ could be a monotonous decreasing function for the same reason as in (a). For best support of the direction change, one can assign the COPU(s) located nearest to the current presentation point a relevance value close to 1. The function $r_{History_S}$ follows the general definition of r_{A_S} introduced earlier.

(d) Working Points and other Presentation Information

The interaction sets defined so far are dynamic, i.e., the sets change as the presentation goes on and the state S=<p, skip> changes. In contrast to these dynamic sets one can define static sets which do not change with the presentation flow. These interaction sets are defined on presentation information, which is static over a presentation cycle. Typically, working points (often also called bookmarks) defined by users constitute such information. For example, in a video edit-

ing workbench, the editor is interested to mark specific points for later cut operations. These points can be made visible to the COM by extending the presentation state s with information about working points. The COM puts a higher relevance value on these points and their environments in order to prevent them from replacement and to allow for their timely preloading. This shortens the presentation delay when such a marked position is reactivated.

Besides explicitly marked working points, one can imagine an automated recognition of content–dependent working points. For instance, scene changes in a video are good candidates for such points as they are likely entry points for presentations. Methods as suggested in [8] could be used for automatic recognition of scene changes for specific data formats. The detection of higher semantic information in videos can not currently be done automatically, but multimedia products such as annotated movies can be taken as a good example for information that can be used by the COM for internal optimization. Another example are traffic news which are preceded by a special signal as currently broadcasted in some countries. Traffic news can be considered in a user–dependent way. Furthermore, it can be expected that in the future more information can be derived from multimedia data, e.g., the recognition of news or commercials [10].

We define an interaction set $Point_s$ able to treat this type of information. It contains all COPUs in the neighborhood of a working point p. It is not sufficient to mark only the working point itself with a high reference, because these points are normally just entry points for further presentations. Moreover, considering an environment around a working point like a scene change allows the COM to efficiently support "fuzzy" access around the scene change. This is especially useful in the case that, e.g., the user interface of a presentation component does not visualize the existence of a scene cut. The fast start of a presentation can still be guaranteed, if a user initiates a presentation "near" (i.e., in the neighborhood of) a scene change he may remember. The minimum number of COPUs determining the neighborhood for a working point c_{wp}, $c_{wp} \in CO$ and $wp = 0, ..., |CO|-1$, is influenced by the tolerable delay of presentation started at point c_{wp}.

For a given working point c_{wp} and its neighborhood $[wp - extleft,\ wp + extright]$, $c_{wp - extleft} \in CO$, $c_{wp + extright} \in CO$ the interaction set $Point_s$ is defined as follows:

$$Point_s = \{ (c_j, dr_{Point_{wp}}(i)) \mid c_j \in CO,\ j = wp - extleft + i,\ i = 0,1,2, ..., (extleft + extright) \}.$$

Note that the calculation of these interaction sets need an extension of s in the way that the parameters wp, extleft and extright for each working point are available. For the function $dr_{Points}(i)$, $0 < dr_{Points}(i) < 1$ has to hold for all i. It can be defined in such a way that, e.g., it has a maximum at $i = extleft$ and monotonously decreases for $i \in [0, extleft]$ and $i \in]extleft, extleft+extright]$. The function r_{Point_s} follows the general definition of r_{A_s} introduced earlier.

If one can predefine the set of operations allowed to be applied at working point c_{wp}, specific predefined information like direction and skip could be considered in the definition of $Point_s$. For example, knowing that only a *fastForward*–operation can be applied at a working point c_{wp}, will lead to $extleft = 0$, and $j = p + 2 \cdot i$. Furthermore, heuristics like a playForward–operation at working points near at the beginning is more probable, can be expressed. Note that the membership to a working point does not change during a presentation, i.e. the interaction set $Point_s$ is static.

General Replacement and Preloading Strategy

Instead of developing two independent strategies for replacement/preloading, we follow a unified approach by using the relevance function r_{CO} for both aspects of buffer management. The reason for that is that the buffer management has to guarantee continuity by providing at least those COPUs in the set $Referenced_s$, which have assigned a relevance value of 1. During a presentation the COM should avoid replacement of COPUs needed at a later time in this presentation and of potentially referenced COPUs by an interaction.

Let us now outline the L/MRP general preloading and replacement algorithm. The L/MRP algorithm is initiated by the COM at every request for getting a COPU to be presented. The next replacement victim during a presentation is the COPU c available in buffer with minimum value $r_{CO}(c)$. The COPU d with maximum value $r_{CO}(d)$ is the next COPU that has to be preloaded if it is not yet present in buffer. In the following algorithm, let $BUFFER$ denote the set of COPUs which are present in buffer.

General L/MRP Algorithm

```
GetNextCopuToBePresented( s ) : Pointer to COPU
begin
(1) for all c ∈ CO with r_CO,s(c) == 1 do
      begin
      // preload the most relevant COPUs
      if c ∉ BUFFER then
         // buffer fault
         if |BUFFER| == <Buffer_Size> then
         // buffer full
            begin
            // replace the least relevant COPU v
(2)         v ∈ {c_j, c_j ∈ BUFFER, r_CO,s(c_j)=min!};
            replace v by preloading c
            end
         else
         // just load c into buffer
            preload c into buffer;
      end
return buffer address of COPU c_p
end
```

The algorithm guarantees that COPU p to be presented next is in buffer, because of $r_{Referenced}(p) = 1$, i.e., COPU c_p is most relevant and will be preloaded, if necessary. In statement (1) of the algorithm for a presentation state s, the COPUs actually checked are $(c_i, r) \in Referenced_s$, where $i = p + h \cdot skip$ with $h = 0, 1, 2, ..., f$, and f denoting the number of COPUS to be prefetched. In statement (2) the distance relevance functions are used to compute the relevance values for |BUFFER| COPUs. The total overhead of the algorithm is O(Buffer_Size). In [15] an optimized implementation of L/MRP with overhead O(1) under the assumption of only monotonous distance relevance function is given.

Variations of the algorithm can take into account that, due to performance reasons, preloading could be performed by means of reasonable quantities of new COPUs with respect to the used underlying storage manager functionality. In general, preloading a sequence of COPUs may be chosen instead of a one-COPU-at-a-time strategy. Then, replacement should also be performed on the basis of bulks of COPUs.

The general L/MRP algorithm is adaptable to application specific aspects in the way that new interaction sets can be integrated easily. For example, a specific application may require specific system support for scanning a whole video. This can be achieved by introducing an interaction set $Scan_s$, which consists of, e.g., every 100th COPU of the video and by defining the distance relevance function dr_{Scan} as, e.g., constant with value 0.8. The definition of the relevance function r_{Scan} follows the definition of r_A introduced earlier, and the definition of r_{CO} has to include the function r_{Scan}. The general L/MRP strategy needs not to be redefined.

4. Preliminary Experimental Results

In the following we introduce two application scenarios with *significantly different* interaction patterns. In some preliminary experimental simulation results we illustrate that L/MRP performs better in these scenarios than some state-of-the-art buffer replacement strategies. The two application scenarios are not taken from some real world applications, but are instantiations of a simulation model. The simulation model allows for definition of two extremely different models in terms of interaction patterns. The important issue is that one scenario represents a highly interactive presentation scenario and the other a less interactive scenario, and, therefore, the behavior of L/MRP against other strategies with respect to interaction can be studied. We will compare L/MRP with the well–known algorithms OPTIMAL, LRU, and RANDOM and then with extended "Use&Toss" strategies.

For the highly interactive presentation scenario we chose a *Video Editing Workbench*–like application, and for the less interactive scenario a *Video on Demand*–like application. By specifying typical presentation parameters like the interval length of an individual presentation of (parts of) a video, skip values, and the probability that specific types of presentations will take place, we can later characterize each scenario. The following simulation model will be used to describe the two scenarios:

- $len = |CO|$, the length of the video.
- $P_{1/100}$, $P_{1/10}$, $P_{1/2}$, P_1 and P_X, the probabilities of typical presentation interval sizes relative to len. For example, $P_{1/100}$ describes the probability that a presentation interval has a length of $1/100 \cdot len$; P_X is the probability that the presentation interval size is $X \cdot$ len, where X is a random number out of]0, 1].

The type of interaction which can occur at the end of a presentation interval can be characterized as follows:

- we consider the presentation states *playForward* (skip=+1), *fastForward* (skip=+2), *playBackward* (skip=–1) and *fastBackward* (skip=–2).
- $P_{playForward}$, $P_{fastForward}$, $P_{playBackward}$ and $P_{fastBackward}$ are the probabilities for the respective operation modes. Clearly, the sum of those must be 1.
- *Continuity probability.* Subsequent presentations are often correlated in the way that the end point of the last presentation is the start point of the next one. This reflects the user behavior of non–random access to other parts of a video as new starting point for the following presentation. For the discussion herein we have chosen a continuity probability of 0.8 for both scenarios to simulate this behavior, i.e., with a probability of 0.8 the user continues at the same presentation point after an interaction. We varied this parameter in order to determine its impact on L/MRP's performance.

In the Video Editing Workbench scenario we have a potentially high degree of short presentations in contrast to the Video on Demand scenario, where primarily playForward–operations of the whole video are undertaken. Fastplay and backward operations are seldom in a Video on Demand scenario. But these can be seen as typical operations performed by an video editor in order to find out, e.g., suitable cut points. We characterize the Video Editing Workbench and the Video on Demand scenarios by instantiations of the simulation model denoted to by VEWB and VoD. They present two types of scenarios which significantly differ in their interaction patterns on a video. For example, the parameter P_1 (the probability that the whole video is presented) is much higher in the Video on Demand scenario than in the Video Editing Workbench scenario. Table 1 lists the parameter settings.

parameter	VEWB	VoD
$P_{1/100}$	0.29	0.0
$P_{1/10}$	0.3	0.025
$P_{1/2}$	0.1	0.025
P_1	0.01	0.85
P_X	0.3	0.1
$P_{playForward}$	0.49	0.81
$P_{fastForward}$	0.21	0.09
$P_{playBackward}$	0.21	0.09
$P_{fastBackward}$	0.09	0.01

Table 1: Instantiations of the simulation model

Our simulator generates presentations on a random basis with respect to the probability distributions defined by the tuple VEWB, VoD, and the continuity parameter. For a given size *len* of a continuous object the total number of touched COPUs in a Video on Demand scenario are significantly higher, due to longer mean presentation intervals. For both scenarios we generated reference strings corresponding to about 10 hours of an interactive video presentation (around 900.000 COPUs). For both of the applications we measured the total number of buffer faults for the whole string.

For the L/MRP strategy we specified the interaction sets *Referenced*, *Skip*, *History* and the working point "begin of the video". The total number of COPUs to be preloaded is 50, i.e. $f = 50$. Assumed that the average loading time of the underlying storage system is at least as much as the presentation speed of an video in PAL–quality, the maximum delay at the beginning of a presentation/after an interaction is 2 sec. We used the already mentioned distance relevance function $dr_{Referenced}$ with $\alpha = 10^{-7}$ (see chapter 3 (a)). For $dr_{History}$ and dr_{Skip} we used the function *f(i)* as follows:

$$f(i) = \begin{cases} 0.9999 - \beta \cdot i, & 0.9999 \cdot \frac{1}{\beta} > i \geqq 0 \\ \beta, & \text{otherwise} \end{cases}$$

We have chosen $\beta = 10^{-3}$ both for dr_{Skip} and $dr_{History}$. This parameter setting reflects the heuristic that referenced COPUs have a 10.000^{th} higher relevance compared to COPUs considered in Skip and History and that the degree of importance for those COPUs is of the same value. The interval of the neighborhood of the working point "begin of the video" in forwardPlay direction was defined to contain 50 COPUs. The relevance values of all COPUs belonging to this neighborhood as well as the relevance of the working point itself are kept near to one (0.9999). Note that these parameter settings have been chosen independently of the simulation model. This concrete parameter configuration can be seen as a *general parameter setting for L/MRP.*

In contrast to L/MRP the classical buffer management strategies are just replacement algorithms and do not consider integrated preloading. A buffer fault occurs every time a requested COPU is not in buffer (*loading on demand*). On the other hand L/MRP guarantees that a requested COPU is always kept in buffer due to preloading. Only a presentation change can cause that the first request of the new presentation can not be satisfied by COPUs in buffer, because of the reestablishment of the preloading area. However, a comparison of an integrated preloading/replacement strategy with a pure replacement strategy by counting the buffer faults caused by consumer requests is not fair, due to the potential risk that the better behavior was just achieved by excessive preloading. For example, a strategy which preloads referenced COPUs, but replaces non–preloaded COPUs with the nearest reference distance would outperform even RANDOM. Consequently, a buffer fault in L/MRP is counted every time a COPU which must be *preloaded* is not found in buffer. Note that the reservation of the preloading area in L/MRP favours the classical strategies as they can fully consider this buffer resource. For example ,if we have a *Buffer_Size* of 10 COPUs and a preloading area of *f*=9, L/MRP can consider just one COPU for better interaction support, whereas e.g. LRU can keep the 10^{th} least recently used COPUs, which probably leads to better results. Obviously, the smaller the preloading area in comparison to the *Buffer_Size*, the less this phenomena is effective.

The OPTIMAL strategy was implemented by first generating all 500 presentations. Every time a COPU has to be replaced, the COPU with the longest reference distance is chosen. This is accomplished by analyzing for every COPU of the buffer, when it will be referenced in a successive presentation and by identifying the COPU as replacement victim which will be presented farthest in the future.

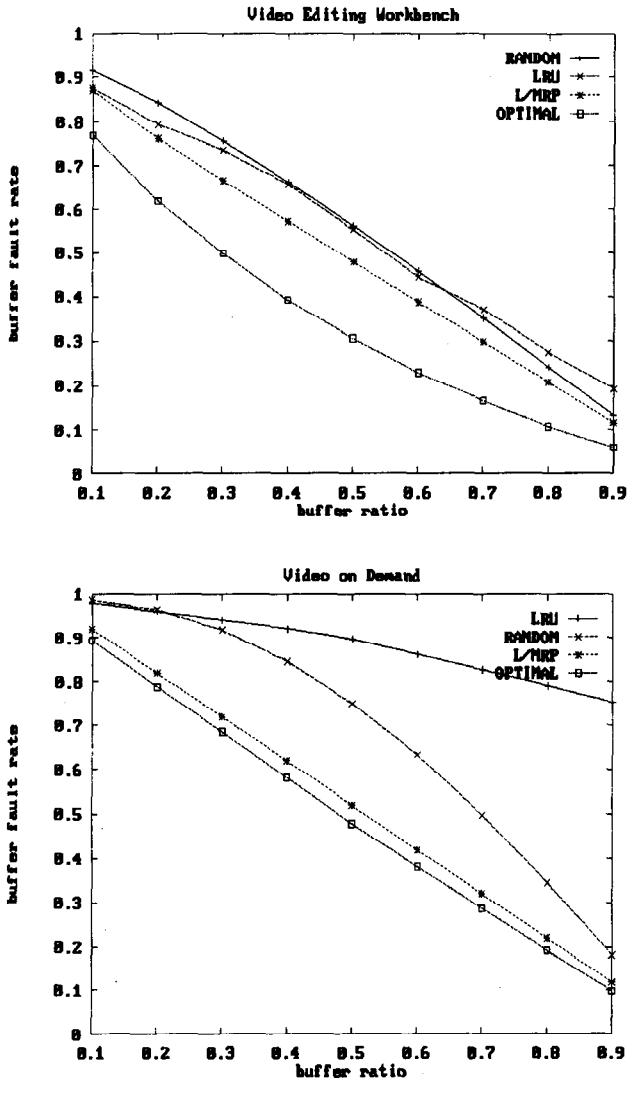

Figure 3: Results for L/MRP. RANDOM, LRU and OPTIMAL

In Figure 3 the results of our simulation for RANDOM, LRU, L/MRP and OPTIMAL are shown. The buffer size was varied between 1/10 and 9/10 of *len* and is displayed on the horizontal axis; the buffer fault rate (total buffer faults divided by total number of touched COPUs during all 500 presentations) of the algorithms are given on the vertical axis.

The figure for the Video Editing Workbench shows a better behavior of L/MRP compared to RANDOM of approximately 5 percent in the average and compared to LRU a better behavior between 3 and 5 percent. The worse behavior to OPTIMAL is between 15 and 20 percent. In the Video on Demand scenario the better behavior of L/MRP compared to RANDOM is between 10 and 20 percent, the gap to OPTIMAL is around 3 percent. The worse and apparently arbitrary behavior of LRU in the Video on Demand scenario might be surprising. The intuitive explanation is that due to the high probability of presentations of the whole video ($P_1 = 0.85$), a destructive behavior of LRU can be observed, similar to the already mentioned example from section 2. Suppose the situation that the whole video is played twice in playForward operation mode. It was evaluated in the simulation that this happens in around 47 percent of the 500 runs. The first play of the whole video results in a buffer state in which all frames nearest to the beginning of the video are least relevant. Hence, they will be replaced successively while the second play is performed. Even for high buffer sizes near to *len* this phenomena can be observed. L/MRP, however, is scan–resistant by keeping always the same group of COPUs in buffer. After the first play of the whole video the buffer is filled by a sequence of COPUs at the end of the video, because replacement victims could only be determined among those COPUs having the longest distance to the end point. When the second play begins a preloading area at the beginning of the video is established by replacing those COPUs which are farthest away. After that, replacement victims are taken from the interaction set History, because the relevance values of them are smaller than the relevances of referenced COPUs, due to the settings of the distance relevance functions.

In quite all multimedia applications a "Use&Toss" strategy is realized, due to the intuitive inefficiency of heuristic buffer strategies. To ensure a fair comparison, in our simulation of "Use&Toss", we do not toss COPUs just after they have been presented, but keep them still in buffer. A direct tossing of the presented COPU would minimize the buffer size, but no interaction support without reloading of COPUs can be achieved. We keep presented COPUs in the remaining buffer part consisting of non–preloaded COPUs. We call this buffer part the *toss area*. A replacement strategy could be defined in the way that replacement victims out of the toss area are chosen on a random basis. It is obvious that this strategy which we call "random Use&Toss" can not perform better than standard RANDOM, because indeed it is just standard RANDOM with preloading. Therefore, we realized also a tougher "Use&Toss" extension, which restricts replacement candidates to non–referenced COPUs in the toss area. Referenced COPUs might be in the toss area, because they could have been "tossed" by preceding presentations. We can express this extended "Use&Toss" in the L/MRP framework just by ignoring the interaction sets *Skip* and *History* and by solely considering the interaction set *Referenced*. This has the effect that all non–referenced COPUs get a relevance of zero. The same distance relevance function $dr_{Referenced}$ as for L/MRP has been used for extended "Use&Toss". This "Use&Toss" extension is advanced in the sense of the general L/MRP–idea of considering interaction semantics. Hence, in addition to a performance comparison between extended "Use&Toss" and L/MRP we can also show how the interac-

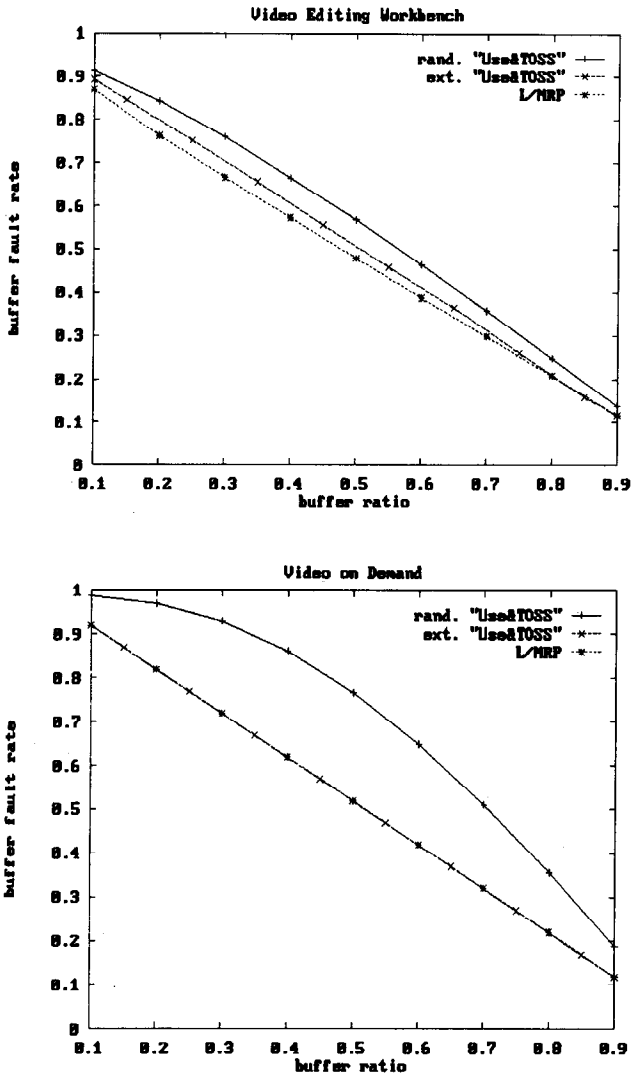

Figure 4: Results for L/MRP and "Use&Toss" variations

tion sets *Skip* and *History* actually influence the overall performance.

In Figure 4 the simulation results of L/MRP, random "Use&Toss", and extended "Use&Toss" for the two scenarios are shown. It can be derived that L/MRP outperforms extended "Use&Toss" with around 4 percent in the highly interactive scenario Video Editing Workbench. In the Video on Demand scenario the two strategies L/MRP and extended "Use&Toss" perform about the same (indeed the curves overlap completely), i.e., the consideration of the interaction sets *Skip* and *History* in less interactive environments with probable long presentation intervals is not effective.

Another remarkable result is that the gap between L/MRP and OPTIMAL in the Video on Demand scenario is less than 3 percent (see Figure 3) and that this gap is constant over the various buffer sizes. It can be outlined that the optimization potential is rather limited in order to find out better strategies than L/MRP or extended "Use&Toss" for this scenario. As expected, the heuristic "never replace a referenced COPU" seems to be sufficient for Video on Demand.

Variations of the continuity probability turned out to have no impact on the relative performance of L/MRP against extended "Use&Toss" for both the Video on Demand and the Video Editing Workbench scenario. Decreasing the probability from 0.8 down to 0.2 just adds the same offset to the buffer fault rates of both strategies. The reason is that decreasing the continuity probability leads to more jumps to random positions within the continuous object. L/MRP as well as extended "Use&Toss" here did not explicitly support such jumps and thus from the perspective of the subsequent presentation the buffer is 'cold' under both strategies. Note that in L/MRP jumps to certain positions can be supported by specifying working points as discussed in subsection 3 (d).

5. Conclusions

In this paper we introduced the L/MRP buffer management strategy. L/MRP is a buffer management framework for continuos object management for various multimedia applications in order to support minimum waiting time after user interactions. By specifying distance relevance functions for different interaction specific sets of presentation units, L/MRP explicitly supports typical interactions and is tunable towards different interaction patterns. Based on the relevances and the presentation state, our strategy replaces units being least relevant for the presentation, both for the running and the subsequent one, and preloads the most important units.

In a performance study we showed that L/MRP outperforms some other buffer management strategies for both less and the highly interactive scenarios, although the parameters of L/MRP were not tuned to reflect the application specific interaction patterns. However, the superiority of L/MRP increases with the frequency of interactions in an application.

The strength of our approach is its adaptability with respect to the following aspects:

- application specific interactions can be considered in L/MRP by classifying respective interaction sets.
- the reflection of application specific interaction semantics can be expressed by means of distance relevance functions.
- openness of L/MRP is achieved by allowing to define any number of interaction sets.
- L/MRP is applicable to a continuous object manager on client as well as on server side. A server can consider different clients (multiple users) by handling the various interaction sets in the same described way.

Our future investigations will especially concentrate on the identification of the most suitable operator to be chosen for the calculation of the overall relevance of a COPU (r_{co} function) and on the determination of suitable distance relevance

functions and their parameter tuning with respect to concrete application scenarios. We believe that this parameter tuning of L/MRP leads to an approximation of a replacement strategy which replaces the COPU with lowest probability of access depending on the presentation state s. We also want to investigate the possibility of learning interaction patterns and their transformation on distance relevance functions, which could result into a self–tuning L/MRP strategy for any application scenario.

Extensions of the L/MRP strategy might be identified in allowing time–dependent interaction sets, which allows to consider user and/or content information, and in considering additional aspects in the state s, e.g., disk characteristics.

The L/MRP strategy was implemented in the continuous object manager of the VODAK object-oriented database system currently extended with multimedia functionality at GMD–IPSI.

Acknowledgements

We thank Peter Muth, Erich J. Neuhold, Thomas C. Rakow and Gerhard Weikum for comments on earlier drafts of this paper and for the many discussions. We also thank Thorsten Weiler for helping to generate the simulation results.

References

[1] C.Y. Chan, B.C. Ooi and H. Lu, "Extensible Buffer Management of Indexes", in Proc. of the18th Int'l Conference of Very Large Date Bases 1992 (VLDB'92), pp 444–454, 1992.

[2] L.A. Belady, "A Study of Replacement Algorithms for Virtual Storage Computers", in IBM System Journal 5, 2/1966.

[3] H.T. Chou and D. DeWitt, "An Evaluation of Buffer Management Strategies for Relational Database Systems, in Proc. of the 11th ACM SIGMOD Conference 1985, pp 127–141, 1985.

[4] Christodoulakis S., N.Ailamaki, M.Fragonikolakis, Y.Kapetanakis, L.Koveo, "An Object Oriented Architecture for Multimedia Information Systems" , in IEEE Data Engineering, 14(3), September 1991.

[5] E.G. Coffman and P.J. Denning, "Operating Systems Theory", Prentice–Hall, 1973.

[6] A. Dan and D. Towsley, "An Approximate Analysis of LRU and FIFO Buffer Replacement Schemes", in Proc. of ACM SIGMETRICS Conf. 1990, pp 143–149, 1990.

[7] W. Effelsberg and T. Haerder, "Principles of Database Buffer Management", in ACM Transactions of Database Systems, 9(9):560–595, 1984.

[8] Fischer S., W.Effelsberg, "Automatic Detection and Management of Video Segments", Technical Report of Praktische Informatik IV, Universität Mannheim, 1994.

[9] J. Gemmel, S.Christodoulakis, "Principles of Delay–Sensitive Multimedia Data Storage and Retrieval", in ACM Transactions on Information Systems, 10(1), Januar 1992.

[10] A. Hampapur, R. Jain and T. Werrymouth, "Digital Video Segmentation", in Proc. of the 2nd ACM International Conference on Multimedia 1994, San Francisco, October 1994.

[11] R. Juahari, M.Carey and M.Linvy, "Priority Hints: An Algorithm for priority–based Buffer Management", in Proc. of the 16th Int'l. Conference on Very Large Data Bases 1991 (VLDB'91), 1991.

[12] T. Johnson and D. Shasha, "2Q: A Low Overhead High Performance Buffer Management Replacement Algorithm", in Proc. of the 20th Int'l. Conf. on Very Large Data Bases 1994 (VLDB'94), pp. 439–450, 1994.

[13] LeGall D.: MPEG, "A Video Compression Standard for Multimedia Applications", Communications of the ACM, 34(4), pp. 46–58, April 1991.

[14] M. U. Kamath, "Continuous Media Sharing in Multimedia Database Systems", in Proc. of the 4th International Conf. on Database Systems for Advanced Applications (DASFAA' 95), April 1995.

[15] A. Kraiß, "An Object Manager for Continuous Data within the ooDBMS VODAK" (in German), in GMD–Studien 256, 1994, Darmstadt.

[16] R. Ng, C. Faloustos and T. Sellis, "Flexible Buffer Management based on Marginal Gains", in Proc. of the 1991 ACM SIGMOD Conference, pp. 397–396, 1991.

[17] R.T. Ng and J. Yang, "Maximizing Buffer and Disk Utilization for News On–Demand", in Proc. of the 20th Int'l. Conf. on Very Large Data Bases 1994 (VLDB'94), pp. 451–462, 1994.

[18] V.F. Nicola, A.Dan and D.M. Dias, "Analysis of the Generalized Clock Buffer Replacement Scheme for Database Transaction Processing", in Proc. 1992 ACM SIGMETRICS Conf., pp35–46, 1992.

[19] E.J. O'Neil, P.E. O'Neil and G. Weikum, "The LRU/k Page Replacement Algorithm for Database Disk Buffering", in Proc. of the 1993 ACM SIGMOD Conference, pp 297–306, 1993.

[20] Papadimitriou C.H., S.Ramanathan, P.V.Rangan, "Information Caching for Delivery of Personalized Video Programs on Home Entertainment Channels", in Proc. of the 1st IEEE Int'l Conf. on Multimedia Computing and Systems 1994, 1994.

[21] Palmer M., S.B. Zdonik, "FIDO – A Cache that learns to fetch", in Proc. ofthe 17th Int'l. Conf. on Very Large Data Bases 1991 (VLDB'91), Barcelona, September 1991.

[22] T.C. Rakow, P.Dettling, F.Moser, B.Paul, "Development of a Multimedia Archiving Teleservice using the DFR Standard", in Proc. of the 2nd International Workshop on Advanced Teleservices and High Speed Communication Architectures (IWACA'95), Springer Verlag, Heidelberg, 1994.

[23] D. Rotem and J. Leon Zhoa, "Buffer Management for Video Database Systems", in Proc. of IEEE Data Engineering 1995, 18, 1995.

[24] G.M Saco and M. Schkolnik, "Buffer Management in Relational Database Systems", ACM Transactions on Database Systems, 11(4), pp 473–498, 1986.