Q-L/MRP: A Buffer Management Mechanism for QoS Support in a Multimedia DBMS

Pål Halvorsen, Vera Goebel & Thomas Plagemann
University of Oslo, UniK - Center for Technology at Kjeller
Granaveien 33, P.B. 70, N-2007 KJELLER, Norway
{paalh, goebel, plageman}@unik.no

In: Proceedings of 1998 IEEE International Workshop on Multimedia Database Management Systems (IW-MMDBMS'98), Dayton, Ohio, USA, August 1998, pp. 162 - 171

Abstract

Multimedia database systems (MMDBSs) have to be capable to handle efficiently time-dependent and time-independent data, and to support Quality-of-Service (QoS). To support continuous playout of time-dependent data, reservations of the limited resources disk I/O bandwidth and network bandwidth have to be combined with appropriate buffer management. Based on the special requirements of the DEDICATION pre-study, i.e., building a cheap prototype system for asynchronous distance education, we have designed the buffer management mechanism Q-L/MRP. Q-L/MRP is a buffer preloading and page replacement mechanism for multimedia applications with heterogeneous QoS requirements. Q-L/MRP extends L/MRP with two features: (1) it supports multiple concurrent users, and (2) it supports QoS with a dynamic prefetching daemon. This dynamic prefetching daemon is able to dynamically adapt to the changes in network and disk I/O load. Furthermore, QoS requirements from the users like frame rate are mapped into the buffer mechanism. Our performance analysis shows that Q-L/MRP is very suitable for the special environment in DEDICATION and outperforms other buffer management mechanisms. Since we have implemented Q-L/MRP in software only, it is also suitable for other multimedia applications on other systems with different hardware configurations and workloads.

1. Introduction

Distributed multimedia applications, like News-on-Demand, digital libraries, and interactive distance learning, seem to be the key to the future information society. Multimedia database systems (MMDBSs) are a suitable platform for these applications if they are capable to handle efficiently large, complex, continuous, and time-dependent data elements such as video, audio, and animations combined with time-independent data elements like pictures, graphics, and traditional data elements like text and integers [15]. Furthermore, it is necessary that MMDBSs support Quality-of-Service (QoS) and are capable to negotiate it with the other elements of distributed multimedia systems, e.g., networks, transport protocols, and operating systems.

The QoS concept is well known in the networking community [18]; it allows applications - respectively their users - to specify their requirements to the system before they are actually using services of the system. Important QoS parameters comprise: reliability, throughput, delay, and delay jitter. For example, a video frame must be presented within a specific time limit in order to avoid delay jitter, i.e., a glitch in the continuous stream. It is generally accepted, that appropriate taxing policies will motivate users to specify exactly the quality they need, instead of simply requesting always the best QoS. The reason for this is simple, the higher the quality the more resources are necessary, i.e., the more expensive the service. Thus, QoS specifications allow to economically manage system resources like network bandwidth, disk I/O bandwidth, CPU time, and main memory; and to make appropriate reservations to assure the requested real-time behavior. For example, the playback of time-dependent data requires to transfer all data to be presented with a guaranteed rate, i.e., throughput, from a local or remote disk into the applications main memory. In this example, disk I/O bandwidth and network bandwidth are very limited resources and bandwidth reservations have to be combined with appropriate buffer management to efficiently support multiple concurrent users/applications.

In the DEDICATION (Database Support for Distance Education) project at UniK, University of Oslo, we face the problem to support the concurrent playout of lectures that have been given in the so-called *electronic classroom* and are stored on a main stream PC with one large disk. The electronic classroom is used at the University of Oslo and other Norwegian Universities for regular courses since four years. It overcomes geographical separations between the different sites by exchanging digital audio, video, and whiteboard information, i.e., transparencies and annotations, in real-time via an ATM-based network. The goal of DEDICATION is to develop a system that allows students to retrieve and playback lectures of the current term. Students should be able to select particular parts and particular media streams of interest, like video, audio, transparencies, or annotations for playout. The system should support for each media stream the QoS requirements of the student. Obviously, the most limited resource in this system is the disk I/O bandwidth of the PC, and due to pragmatical and economical reasons in our pre-study, we are currently not able to avoid or at least to reduce this bottleneck. However, if the pre-study delivers promising results, we might in the future use a large server with a disk array. Thus, our main concern is to optimize the buffer management to achieve maximal utilization of the available I/O bandwidth.

In this paper, we present an extension of the buffer management mechanisms L/MRP (Least/Most Relevant for Presentation) [11], called Q-L/MRP (QoS-L/MRP), that has been designed in DEDICATION to meet the above mentioned requirements. Q-L/MRP supports heterogeneous QoS requirements of multiple concurrent users and is able to adapt itself to the characteristics of disk and network I/O. For the evaluation of Q-L/MRP, we apply the particular workload of the electronic classroom and the restrictions of DEDICATION. Therefore, we describe briefly the electronic classroom and the DEDICATION project in Section 2. Afterwards, we address in Section 3 the problem of buffer management in MMDBS and we analyze the suitability of known buffer management mechanisms for DEDICATION. Section 4 describes L/MRP and Q-L/MRP. In Section 5, we evaluate the performance of Q-L/MRP and compare it with other buffer management mechanisms. Furthermore, we analyze the general applicability of Q-L/MRP, i.e., without the stringent limitations of DEDICATION. Finally, we summarize and conclude this paper in Section 6.

2. Distance Education Scenario

Distance education refers to all types of studies in which students are separated by space and/or time. The electronic classrooms [1] at the University of Oslo (see Figure 1) overcome separation in space by exchanging digital audio, video, and whiteboard information between two sites of the University of Oslo and one of the University of Bergen. Since 1993, the electronic classrooms are regularly used for teaching graduate level courses as well as for research on QoS support in distributed multimedia systems [14].

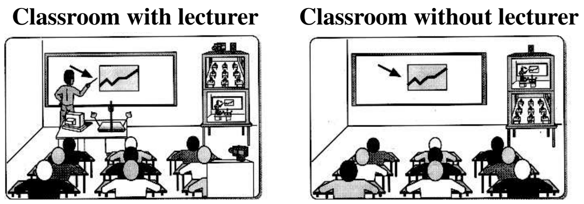

Figure 1. The electronic classroom today.

The main parts of each electronic classroom are:

Electronic whiteboard (EW): at each site there is at least one electronic whiteboard (100") that is used to display lecture notes and transparencies written in Hypertext Markup Language (HTML) format. Transparencies consume about 200 KB (per transparency) and must be kept in the buffer while displayed at the whiteboard. When a section is displayed, the lecturer can write, draw, and erase comments on it by using a lightpen.

Document camera: can be used from the whiteboard application to capture the contents of printed materials, e.g., a page of a book, and present it on the whiteboard.

Audio system: microphones are mounted evenly distributed on the ceiling in order to capture the voice of all the participants. Audio is PCM encoded and is digitized using a 16 bits/16 MHz sampler, which results in a constant data stream of 32 KB/s.

Video system: one camera focuses on the lecturer, and two further cameras focus on the students. A video switch selects the camera corresponding to the microphone with the loudest input signal. Two monitors are placed in the front and two monitors are placed in the back of each classroom displaying the incoming and outgoing video information. A H.261 codec is currently used to digitize and (de-)compress video data.

Table 1 summarizes the used coding and throughput requirements of the different data streams, e.g., transparencies, video frames, audio samples [19].

Media type	Coding	Bandwidth requirement
Video	H.261	Max: 1950 Kbit/s, Min: 13 Kbit/s, Avg: 522 Kbit/s
Audio	PCM	32 KB/s
EW	HTML	200 KB / transparency (loaded once)

Table 1. Workload of electronic classroom

Today, only synchronous teaching is supported, that means the lectures are transfered in real-time over an ATM-based network to the peer classroom(s) and vice versa. Consequently, all students have to be physically present in one of the classrooms during a lecture. In the DEDICATION project, we extend the functionality of today's electronic classroom to support asynchronous teaching by using a MMDBS to store the lectures for graduate level courses. To allow maximum flexibility, all the data types are stored independently. In such a system, students are able to retrieve via network lectures at different times of the day, may search for interesting topics, and playback only parts of lectures. Depending on the student's end-system, network connections, and requirements of the students, different QoS specifications have to be supported. For example, one student might work at home and is connected via ISDN, i.e., 2 $\times$ 64 Kbit/s, to the server. The student has followed the lecture, has a hardcopy of the transparencies, and wants only to recapitulate the explanations of the teacher. Thus, he/she retrieves the audio stream of the particular lecture with maximum quality and the video stream with low quality, i.e., low frame rate. Another student might have missed the lecture and retrieves the full lecture, i.e., audio, video, whiteboard, and document camera, in maximum quality from a terminal that is connected via Fast Ethernet, i.e., 100 Mbit/s, to the server. Generally, we denote situations in which different users potentially have different requirements as heterogeneous QoS requirements.

In the current phase of DEDICATION, we develop a first prototype of this system and use quite cheap equipment: the *test* server[1] runs on a PC with one large disk. Based on the fact that in average 15 students follow a graduate course, we assume generally maximal three concurrent users at a time. This workload can be handled by the PC. However, we are only able to store always the last two stored lectures. Depending on the experiences with the first prototype, its acceptance, etc. we might later on extend the system with a RAID storage system.

For our performance evaluation in Section 5, we consider the workload that is given by the electronic classroom. The only exception is the compression format of video data, because H.261 is designed for real-time transmission and is not appropriate for storing video data [18]. From the appropriate compression techniques like JPEG, MPEG, MPEG-II we have chosen M-JPEG (Motion-JPEG) in order to perform a worst case analysis. This results in a bandwidth requirements of about 1.5 MB/s for the video and about 360 KB/s for the document camera, i.e., about 60 KB per M-JPEG frame. The bandwidth requirements for audio and the electronic whiteboard are described in Table 1. Furthermore, we apply in our simulation model the characteristics of the Seagate Elite 23 disk, which are summarized in Table 2.

[1] Used for testing and simulating buffer mamagement. A real MMDBS server would run on a more powerful machine.

Formatted capacity	23,2 GB
Cylinders	6876
Tracks per cylinder	28
Blocks per track	235
Block size	512 Bytes
Track-To-Track Seek (read/write)	1,1 ms
Average Seek (read/write)	13 ms
Maximum Seek (read/write)	28 ms
Spindle Speed	5400 RPM
Average Latency	5,56 ms
Internal Transfer Rate	10,75 - 15,5 MB/s

Table 2. Specification of the SEAGATE ELITE 23 **Disc Drive [17].**

3. Buffer Management for Multimedia DBS

Until quite recently, work on buffer management addressed traditional applications with only small, time-independent data. Well-known page replacement algorithms like RANDOM, LRU (Least Recently Used), FIFO (First In First Out), LFU (Least Frequently Used), CLOCK, GCLOCK (Generalized CLOCK) and LRD (Least Reference Density) [3] are implemented in most traditional systems. These algorithms as well as their improvements, e.g., LRU-K [13] and 2Q [6] which are extensions of LRU, are not suitable to support time-dependent data in MMDBS. There are two reasons for the inappropriateness of these traditional page replacement algorithms:

- they consider as page replacement criteria only age and reference frequency of pages, and
- they use demand paging.

Time-dependent data requires real-time behavior in terms of the QoS parameters throughput, delay jitter and response time from the MMDBS. The following example demonstrates that demand paging is a poor choice to support time-dependent data, especially when limited delay and delay jitter is required. To simplify the example, we assume a single-user of the distance education application described in Section 2. To playout video frames with a frequency of 24 frames/sec we have an upper bound of delay to read a single video frame of 41 ms, because this is the latest time the next frame has to be available for presentation. For playout applications, demand paging algorithms fetch one frame per period from disk. However, the average latency to read a video frame from disk (with a page size of 8 KB) is approximately 61 ms.

Figure 2 presents calculated transfer times in single-user mode using only average seek time, spindle speed and average latency of the Seagate Elite 23 disk in a demand paging scenario with the previously described requirements from our distance education application. The horizontal line denotes an upper time limit of 41 ms to fetch each presentation unit, and the curves show the calculated transfer times when retrieving different combinations of data types for different page sizes. As we can see, the transfer times of both audio and video are far above the upper limit, so there is no possibility, independent of the page size, to support continuous presentations of an entire lecture with all data types in a demand paging system. Obviously, it is impossible to support multiple concurrent users in this scenario.

The solution of this problem lies in the characteristics of the application: playback of continuous data streams such

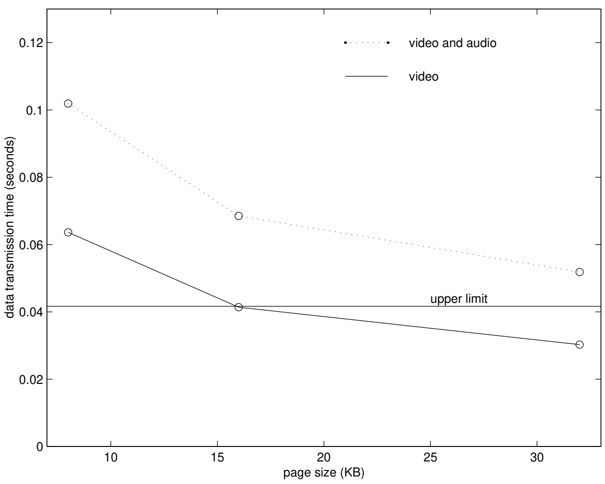

Figure 2. Data transmission time from disk to buffer in a single-user demand paging scenario.

as video is predictable, i.e., data elements are read sequentially. Thus, prefetching of data combined with contiguous data placement on disk reduces the number of page faults and improves I/O bandwidth utilization.

In the last few years, there has been quite a lot of work in the area of buffer management for MMDBSs. Rotem and Zhao [16] compare different ways of partitioning the buffer in order of maximize the number of concurrent users in a Video-on-Demand (VoD) scenario. Ng and Yang [12] address the problem of buffer sharing among multiple users and how to prefetch data from disk to buffer. Liskov et al. [8] present a novel adaptive prefetching strategy for the object-oriented DBS Thor designed for small objects (average 51 bytes). This algorithm does not take into account application semantics, knowlege of typical access patterns, network traffic, and QoS aspects. Furthermore, Moser, Kraiß and Klas [11] present the buffer replacement strategy L/MRP (Least/Most Relevant for Presentation), especially designed for interactive continuous data streams like in VoD scenarios. Hollfelder, Kraiß and Rakow [5] present a smooth buffer adaption technique at the client side which provides an implicit QoS support for playback of time-dependent data streams. Kamath, Ramamritham and Towsley [7] use media caching for data reuse by subsequent users, and they look at the benefits of batching concurrent users and sharing data from continuous media. Dan and Sitaram [2] introduce a caching strategy, GIC (generalized interval caching), to guarantee continuous delivery of time-dependent data. However, these works on buffer management for MMDBSs usually address support of continuous presentation of time-dependent data with homogeneous QoS requirements, e.g., all users request the same frame rate. Furthermore, almost all these papers describe work on large Media-on-Demand servers with almost only pure linear playbacks, e.g., VoD or News-on-Demand (NoD).

All these servers have far more resources than we have on our PC in DEDICATION, e.g., in [2] they use buffers up to a couple of GB whereas we use buffers of 16 MB to 128 MB. Furthermore, the number of users of our application is limited to the number of students, and the inter-arrival time between requests might therefore be in magnitude of hours. We want to support interactive presentation of a lecture where a user can explicitly describe his/her QoS requirements. Most buffer management mechanisms do not explicitly address the problem of supporting heterogeneous QoS requirements like we do for our interactive, distance education application.

We have extended the L/MRP [11] preloading and data replacement strategy with QoS support both on server and client side, denoted Q-L/MRP. The next sections describe L/MRP and Q-L/MRP in detail and show performance evaluations. We demonstrate how useful QoS specifications are to minimize the number of page faults and the amount of data to be loaded from disk to buffer.

4. L/MRP and Q-L/MRP

In this section, we describe both the original L/MRP algorithm [11] and our extensions to support different QoS requirements of different users and to adapt it for our application.

4.1. L/MRP

L/MRP [11] is a buffer management strategy which considers presentation specific information in order to provide an optimized behavior with respect to requirements like continuous presentation of time-dependent data in both pure linear presentations and after frequent user interactions.

The main goal of L/MRP is to replace those data which will not be used for the longest time in the future and to prefetch data into the buffer before they are requested. Based on relevances and presentation state, this strategy replaces units being least relevant for the presentation and preloads the most important units.

L/MRP splits a *continuous object* (CO), e.g., a video clip, into smaller *continuous object presentation units* (COPUs), e.g., video frames. An element c_i, i = 0, ..., |CO| - 1, denotes the COPU with index i within the CO. The state of a presentation is characterized by a tuple s = <p, skip> where p denotes the COPU index at the current presentation point and skip denotes the skip value.

Replacement of data is done by giving each COPU a *relevance value*, and the COPU with lowest relevance value is replaced. These relevance values are assigned by including the COPUs in *interaction sets*, and the reference value is set according to which interaction set the COPU is member of and where the presentation point is in the CO (see Figure 3). Basically, there are three[2] types of interaction sets: (1) *Referenced* - COPUs to be presented in the future, (2) *History* - COPUs in the reverse direction, i.e., COPUs that have been presented, and (3) *Skipped* - COPUs to be skipped, e.g., in a fast forward presentation. Furthermore, L/MRP supports different presentation modes like fast forward, e.g., display every second video frame, by setting a lower reference value to the presentation units not to be displayed.

To denote the relevance of a given COPU within an interaction set, a *distance relevance function*, dr_A(i), is defined in each interaction set (A) which calculates the relevance value for a given COPU according to the COPU's distance (i) from the current presentation point. This means that the distance relevance function describes the degree of importance to keep a COPU in the buffer, i.e., COPUs near the presentation point have high relevance values and are not to be replaced while COPUs far away from the presentation point have a low relevance values and may be replaced. An interaction set A_s for a presentation state s is now defined as a set A_s={$(c_j$, dr_A(i)) | c_j ∈CO, i∈N_0, j = g(i,s)} where the

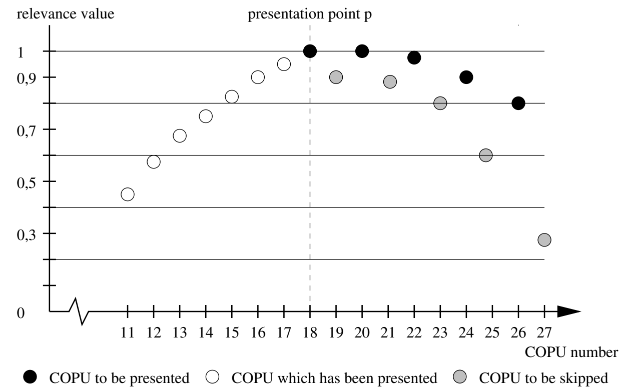

Figure 3. Reference values in L/MRP [11].

index j of the COPUs to be considered is determined by a function g which depends on the current presentation point p and the distance i to p. To compare the relevance values of all COPUs, a *relevance function*, r_{A_s} is defined as

$$r_{A_s} : CO \rightarrow [0, 1]$$
$$r_{A_s}(c) \;=\; \begin{cases} \text{v}, & (\text{c, v}) \in \text{A}_s \\ 0, & \text{otherwise} \end{cases}$$

where the COPU's relevance value for a given interaction set is equal to the distance relevance value or NIL according to whether the COPU is a member of the set or not, respectively. Furthermore, a COPU can be considered in multiple interaction sets. In order to determine the overall relevance of a COPU with respect to all the interaction sets $A_{k,s}$ the relevance function for all COPUs in CO is defined as

$$r_{CO,s} : CO \rightarrow [0, 1]$$
$$r_{CO,s}(c) \;=\; \max_{k=A_{1,s},\cdots,A_{n,s}} (r_k(c))$$

where r_k calculates the COPU's relevance value according to interaction set k. r_{CO} then assigns the maximum of these relevance values to the COPU.

Replacement and prefetching of data is integrated into one general L/MRP algorithm (see Figure 4). This algorithm is initiated each time a COPU is requested for presentation. Then all the COPUs which are most relevant for presentation, i.e., those COPUs where $r_{CO,s}$(c) = 1, are prefetched into the buffer. If there is some free buffer space, these COPUs are just loaded into the buffer. Otherwise, the COPUs with a minimum value $r_{CO,s}$(d) are the replacement victims, and the COPUs with the least relevance values are replaced by preloading the COPUs with the highest relevance values.

GetNextCopuToBePresented(s) : Pointer to COPU
  for all c ∈ CO with $r_{CO,s}$(c) = 1 **do**
    if c ∉ BUFFER **then**
      if buffer is full **then**
        v∈{c_j | c_j,c_k ∈BUFFER,$r_{CO,s}(c_j)$=$\min_{k=0,\ldots,|CO|-1}(r_{CO,s}(c_k))$}
        replace v by preloading c
      else
        preload c in a free buffer space
return buffer address of COPU c_p

Figure 4. General L/MRP algorithm [11].

[2] More sets can be added, see [11].

4.2. Q-L/MRP

We have extended L/MRP to support our distance education application by defining different interaction sets with different distance relevance functions. Q-L/MRP supports not only one data stream, but all the data streams in our scenario, i.e., a stream for the electronic whiteboard, a video stream, an audio stream, and a stream for a document camera. Furthermore, L/MRP is designed for a single-user environment using small buffers containing some seconds of a M-JPEG video. It is implemented in the continuous object manager of the VODAK object-oriented database system [11] where it runs on the client side in a client/server environment. We test Q-L/MRP both at the server and the client side for our application with support for multiple users. The two extensions are quite trivial and are provided by adding additional interaction sets to each data stream and each user.

To support users with different QoS requirements, we map QoS specifications into the buffering mechanism. For example, if a user wants to reduce the frame rate of the video presentation, e.g., to half of the current frame rate, Q-L/MRP loads only every second video frame, i.e., drops the others frames, and thereby increases the available system bandwidth. L/MRP would always load all the frames regardless of the user request.

```
if total time waiting for I/O ≤ 0.2 s then
    amount = 1 s; /* PREFETCH THE next second OF DATA every 250 ms */
    timer = 0.25 s;
else if 0.2 s < total time waiting for I/O ≤ 0.45 s then
    amount = 2 s; /* PREFETCH THE next 2 seconds OF DATA every 500 ms */
    timer = 0.5 s;
else if 0.45 s < total time waiting for I/O ≤ 0.7 s then
    amount = 3 s; /* PREFETCH THE next 3 seconds OF DATA every 750 ms */
    timer = 0.75 s;
else if 0.7 s < total time waiting for I/O ≤ 0.95 s then
    amount = 4 s; /* PREFETCH THE next 4 seconds OF DATA every 1000 ms */
    timer = 1 s;
else if 0.95 s < total time waiting for I/O ≤ 1.2 s then
    amount = 5 s; /* PREFETCH THE next 5 seconds OF DATA every 1250 ms */
    timer = 1.25 s;
else if 1.2 s < total time waiting for I/O ≤ 1.45 s then
    amount = 6 s; /* PREFETCH THE next 6 seconds OF DATA every 1500 ms */
    timer = 1.5 s;
else
    amount = 7 s; /* PREFETCH THE next 7 seconds OF DATA every 1750 ms */
    timer = 1.75 s;
```

Figure 5. Configuration of the preloading daemon.

In section 3, we outlined the importance of prefetching data from disk to buffer in order to be able to retrieve data in time for continuous playback of time-dependent data streams. Figure 2 shows that due to performance reasons, a one-COPU-at-a-time preloading strategy is not adequate for our application. Instead of activating the L/MRP buffer management strategy each time a COPU is requested for presentation, we create a prefetching daemon which is activated by a page fault or after a given time interval. If the total execution time of the daemon including waiting for I/O exceeds the given time interval, the daemon is activated again whenever the previous process is finished[3].

How much data has to be preloaded and with which preloading frequency is strongly dependent of each other. A high frequency (many requests) introduces additional round trip delays and disk accesses and might cause more page faults. If the amount of data to be preloaded each time is too high, we might waste bandwidth on loading data which will not be used, e.g., the user interacts with the presentation. Thus, support of continuous presentation of time-dependent data in all types of I/O system and network workloads, i.e., reduce the number of page faults, in single or multi-user mode, and heterogeneous QoS requirements, and optimizing the available bandwidth of the I/O system, i.e., do not preload data not needed, can only be provided via dynamic adaption of the buffer management mechanism. We have created a dynamic prefetching daemon which tries to balance these contradictionary parameters, i.e., frequency and amount of data. The frequency and data amount is dynamically adjusted according to the system workload (total waiting time for I/O). A dynamic daemon configuration example is shown in Figure 5. For implementation details, see Section 5.

The amount of data to be preloaded and the preloading frequency is mutually dependent. A high frequency results in many requests, each for a small amount of data. Vice versa, low frequency results in less requests for larger amounts of data. The choice for a particular preloading frequency and the particular amount of data to be preloaded is depending on: available disk I/O bandwidth, network load, QoS requirements, and other factors. For example, the higher the total transfer time from disk the better is a low preloading frequency. However, the larger the amount of data to be preloaded, the higher the potential waste of bandwidth by preloading the wrong data. With respect to the fact that most of the influencing factors are dynamic, it is obvious, that only dynamic adaptation of the buffer management mechanism will assure an optimal solution.

We have created a dynamic prefetching daemon which tries to balance these contradictionary parameters, i.e., frequency and amount of data. The frequency and data amount is dynamically adjusted according to the system workload (total waiting time for I/O). A dynamic daemon configuration example is shown in Figure 5. For implementation details, see Section 5.

5. Performance Evaluation

In order to be able to evaluate the performance of the buffer management mechanism without any side effects from other system components and to be able to use exactly the same environment each time, we have implemented and simulated Q-L/MRP in *Matlab* [9]. Matlab is a mathematical modeling tool for simulations and the programming language is very similar to C or Pascal. We run the simulations using homogeneous QoS requirements. We assume a system environment and application requirements as they are described in Section 2.

As simulation data, we have taken a five minutes long, randomly chosen time window from a stored lecture which is representative for an entire lecture. We use two reference strings for the playback of this lecture based on [14]: (1) pure linear playback, and (2) playback including playback forward, backward, in other speeds, and jumps backward and forward. For simulating Q-L/MRP on the client side, we use three different network round trip delay figures as they are illustrated in Figure 6. These figures are based on measurements we have performed in the target environment of DEDICATION [19], i.e., between UniK and the main campus of the University of Oslo (a distance of about 30 km). We use these figures to analyze how Q-L/MRP adapts

[3]We have two choices regarding daemon activation: (1) activate whenever the previous daemon finishes (as described in this paper), or (2) activate after a given time interval regardless whether the previous process has completed its work or not. We have chosen the first alternative, because we did not want to have multiple daemon processes which might request the same data from disk (multiple transmissions of the same data).

to changes in the total transmission time introduced by the network.

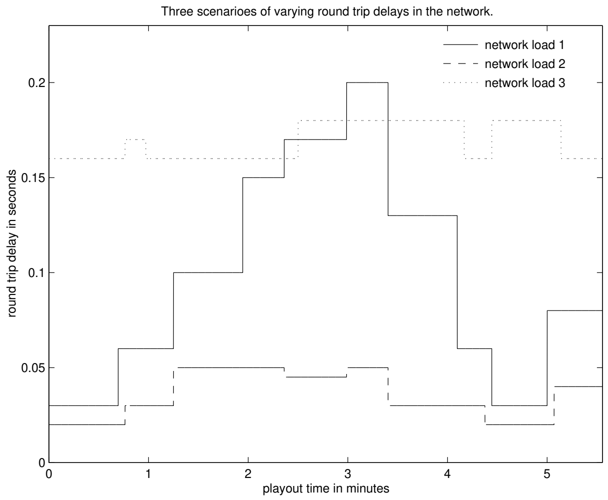

Figure 6. Different round trip delays over the network.

In order to evaluate if Q-L/MRP can support continuous presentation of time-dependent data we analyze the number of page faults and delays generated by Q-L/MRP. We simulate three different scenarios:

1. A centralized system where the client first downloads the entire lecture.
2. A distributed system where the client fetches data from a remote server (Q-L/MRP at the client side). The client controls the playout and sends a request to the server each time more data is needed (*pull approach*). This scenario is divided into two parts: (A) data must be retrieved from disk at the server, and (B) data is already loaded into the server buffer.
3. A distributed system where clients fetch data from a remote server (Q-L/MRP at the server side). The client handles the playout but sends only requests concerning interactions, e.g., start, stop, jump etc. The server transmits data to the client (*push approach*).

We run several simulations to prove that Q-L/MRP increases the system throughput by generating less page replacements than traditional algorithms. We compare Q-L/MRP with LRU and RANDOM for two reasons: (1) other related work, which the authors know about, do not address the problem of heterogeneous QoS specifications, do not directly address data preloading and page replacement, and are designed for much larger environments than our distance education application, and (2) LRU and RANDOM are widely used in traditional buffer management mechanism in database management systems (DBMSs). At the client side, we also compare Q-L/MRP to an extended version of L/MRP (support of all our data streams). The different simulation scenarios are based on the described bandwidth requirements and reference strings. Due to the particular requirements and the workload of our application, we simulate single-user mode and multi-user mode with three concurrent users. We also compare page replacement using different preloading strategies, and analyze the gain of data reuse[4] by increasing the buffer size. The following subsections present the simulation results.

```
for each media type do
  /* DISTANCE RELEVANCE FUNCTIONS: */
  /* - THE Referenced INTERACTION SET */
  dr_Ref(α, dist, amount) = { 1,               0 ≤ dist ≤ amount
                            { 1 − α × dist,    amount + 1 ≤ dist

  /* - THE History INTERACTION SET */
  dr_His(β, dist) = 1 − β × dist

  /* - THE Skipped INTERACTION SET */
  dr_Skip(γ, dist) = 1 − γ × dist

  /* COMPUTE DISTANCE n BETWEEN TO COPUs IN REFERENCED*/
(1)if skip > 0 then
     n = skip + rate - 1
  else
     n = skip - rate - 1
  end

  /* COMPUTE AND ASSIGN THE RELEVANCE VALUES */
  /* TO THE COPUs IN THE BUFFER */
(2)for each COPU c ∈ BUFFER where COPU c is of type media type do
     /* COMPUTE DISTANCE d FROM COPU j TO PRESENTATION POINT p */
     d = j - p

     if (d < 0 && skip > 0) || (d > 0 && skip < 0)
       /* COPU ∈ History */
       relevance value of COPU c = dr_His(β, d)
     else if mod(p, n) = mod(p + d, n)
       /* COPU ∈ Referenced */
       relevance value of COPU c = dr_Ref(α, d, amount_to_preload)
     else
       /* COPU ∈ Skipped */
       relevance value of COPU c = dr_Skip(γ, d)
     end
  end

  /* PRELOAD ALL COPUs WHOSE RELEVANCE VALUE IS ONE */
(3)for count = 1 : amount_to_preload; step rate do
     i = p + count × n
     find COPU_k with lowest reference value
     replace COPU_k with COPU_i
  end
end
```

Figure 7. Pseudo code of the prefetching daemon.

The implementation of Q-L/MRP for the *client* side in our simulation environment is outlined in Figure 7 with simplified pseudo code. On the *server* side, points (1), (2), and (3) have to be executed for each user as well. The algorithm in Figure 5 determines the amount of data to be preloaded and the execution frequency. Table 3 shows the respective parameters. If we do not have straight playback forward, the audio stream is not presented, and the *rate* has only meaning in the video and camera case, i.e., rate is set to one for audio. Saving and loading registers etc. during a context switch is assumed to take about 50 micro-seconds based on measurements for an IBM PowerPC in [10].

5.1. Support for Continuous Presentation of Time-Dependent Data

In order to see if Q-L/MRP can support continuous presentations of time-dependent data, we present simulation results for all the different system scenarios described above with buffer sizes of 32 MB and 64 MB, page sizes of 8 KB, 16 KB, and 32 KB, and the two reference strings.

[4] By the term *data reuse* in this context, we mean that the data needed is still in the buffer. An improved data reuse gives less page faults and a smaller amount of data must be loaded into the buffer.

Parameter	Video	Audio	Camera	Transparencies
α	$\frac{1}{14400}$	$\frac{1}{10}$	$\frac{1}{3600}$	$\frac{1}{2}$
β	$\frac{1}{720}$	$\frac{1}{30}$	$\frac{1}{180}$	$\frac{1}{2}$
γ	$\frac{1}{720}$	$\frac{1}{30}$	$\frac{1}{180}$	-

Table 3. Distance relevance function parameters used in our simulations.

In the first scenario (centralized, isolated system), the results are similar to those presented in [11]. Q-L/MRP reduces the number of page faults dramatically compared to traditional, demand paging algorithms like RANDOM and LRU, i.e., in all buffer and page size configurations, we have only one page fault in the interactive reference string which results in a delay of about 0.06 seconds for presenting data on the electronic whiteboard. The pure linear playback is presented without any page faults.

Simulating Q-L/MRP at the server with three concurrent users, pure linear playback is presented with one or two page faults giving delays of about 0.3 seconds. Using the interactive reference string, we get between 171 and 174 page faults resulting in delays from 0.15 to 0.59 seconds. Compared to the total amount of pages fetched from disk, we have about 0.2% page faults depending of the buffer size using 16 KB pages, and compared to demand paging mechanisms this is a large improvement as shown in Table 4. The prefetching mechanism in Q-L/MRP produces only 0.09 - 0.37% of the number of page faults compared to RANDOM and LRU in a 32 MB or a 64 MB buffer.

Buffer size (MB)	*Page size (KB)*	*Algorithm* Q-L/MRP	LRU	RANDOM
32	**8**	173	185290	190124
	16	172	92645	95065
	32	172	46325	47576
64	**8**	174	185240	185939
	16	171	92620	92916
	32	172	46312	46513

Table 4. Number of page faults in a server with three concurrent users.

Additionally, we simulate L/MRP and Q-L/MRP on the client side. Our results shows that the algorithm execution time and the time to retrieve all data from disk and transmit it over the network is too high to support continuous presentation of a lecture using L/MRP. Before one request is finished another two to six requests has arrived depending on page size, buffer size and network delays. This means that the L/MRP must run each request in parallel (which will be many) and use a some extra CPU-time, or if there are only one request run at a time, there will be various number of page faults resulting in delays in the presentation.

Simulations using Q-L/MRP and no network delays give the same results as in the first scenario, i.e., at most two page faults. Simulating with the network delays shown in Figure 6, we compare our dynamic preloading mechanism (see Figure 5) which adapts to changes in the network and disk I/O (denoted Q-L/MRP$_{\text{dyn}}$) with two static preloading strategies (denoted Q-L/MRP$_{\text{min}}$ and Q-L/MRP$_{\text{max}}$). These static preloading strategies are similar with respect to adaption to an extended version of L/MRP which runs a prefetching daemon which is statically configured like the one originally implemented in L/MRP. The two static strategies are configured as *high preloading frequency with a small amount of data to prefetch*, e.g., prefetch the next second of data every 250 ms, and *low preloading frequency with a large amount of data to prefetch*, e.g., prefetch the next seven seconds of data every 1750 ms. Compared to demand paging algorithms, Q-L/MRP provides a large improvement in the amount of page faults.

Network load/ Buffer size	*Algorithm* Q-L/MRP$_{\text{dyn}}$			Q-L/MRP$_{\text{max}}$			Q-L/MRP$_{\text{min}}$		
Load 1/32 MB	5	5	5	261	261	261	28	26	27
Load 2/32 MB	5	5	5	261	261	261	4	4	4
Load 3/32 MB	4	4	4	261	261	261	160	54	54
Load 1/64 MB	5	4	4	261	261	261	27	27	28
Load 2/64 MB	5	5	5	261	261	261	5	5	5
Load 3/64 MB	7	4	4	261	261	261	191	54	54
Page sizes (KB)	8	16	32	8	16	32	8	16	32

Table 5. Number of page faults depending on preloading strategy using an interactive reference string.

From Table 5, which shows the simulation results where data has to be retrieved from the server disk, we conclude that a dynamic preloading strategy supports continuous presentation of time-dependent data better than static ones. When data is found in the server buffer, Q-L/MRP$_{\text{max}}$ generates exactly the same amount of page faults. However, since we do not have to retrieve data from disk, the total waiting time for I/O decreases. In this simulation scenario, Q-L/MRP$_{\text{dyn}}$ is configured by the system exactly as Q-L/MRP$_{\text{min}}$. Both generate almost the same results as Q-L/MRP$_{\text{dyn}}$ does in the case where the disk has to be accessed. Furthermore, our simulation results show that the implemented prefetching and page replacement mechanism in Q-L/MRP supports continuous presentation of lectures from our distance education application, and the response time after interactions varies from 0 to 0.70 seconds, which is below PAL's (phase alternating line) two seconds restart delay requirement [11].

5.2. Amount of Data Loaded into the Buffer

In this Section, we compare the amount of data loaded into the buffer from the page replacement strategies Q-L/MRP, L/MRP, LRU, and RANDOM. We present only the multi-user mode results using pure linear playback, because there is nearly no reuse of data in a single-user situation, the results are almost identical for all the page replacement algorithms, and similar results are presented in [11]. Using playback with interaction, we present the results from single-user mode. We compare the dynamic and static preloading strategy, and present the benefits of increasing the buffer size.

5.2.1 Comparison of Page Replacement Strategies

The results from the multi-user simulation (see Figure 8) with a 32 MB buffer show that Q-L/MRP gives better performance by reducing the amount of data loaded into the buffer. This is a large improvement, e.g., in case of 8 KB pages, Q-L/MRP, LRU, and RANDOM load 1406.9 MB, 1447.6 MB, and 1485.3 MB of data respectively. Similar results are achieved with the 64 MB buffer simulation, but the amount of loaded data is further reduced compared to LRU and RANDOM. If we look at the referenced data, Q-L/MRP preloads more data than necessary when using a 32 MB buffer, but increasing the buffer size to 64 MB results in more data reuse, and the total amount of loaded data is smaller than the amount of referenced data.

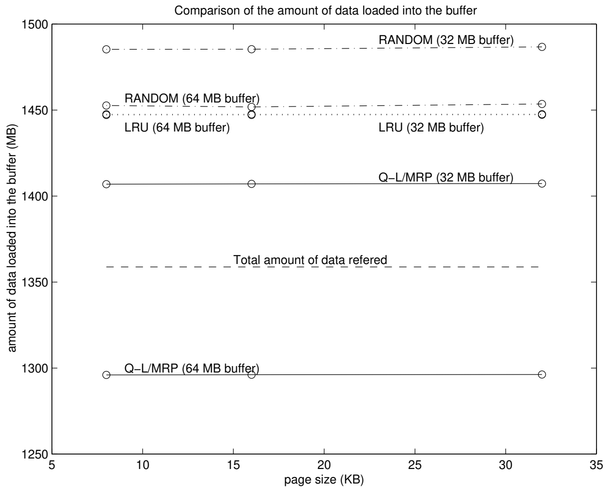

Figure 8. Page replacements for different paging strategies (non-interactive scenario, three users).

Like the simulation without interactions, Q-L/MRP gives the best results in the single-user playback simulation with all kinds of user interactions. Both the 32 MB and the 64 MB buffer simulation show an even larger improvement compared to LRU and RANDOM than in the non-interactive scenario (see Figure 9, Q-L/MRP and L/MRP have almost overlapping curves). Compared to L/MRP, the gain in the amount of loaded data is very low, i.e., Q-L/MRP loads about 506.1 MB while L/MRP loads 507.1 MB using a 32 MB buffer. If we again compare with the referenced data, Q-L/MRP loads less data than referenced no matter what buffer size we use.

Furthermore, if we compare Q-L/MRP (which maps user requirements) and L/MRP (which always loads the maximum amount of data), the amount of data loaded is reduced by the number of COPUs not loaded multiplied by the COPU size. For example, if the user in a pure linear playback reduces the frame rate to the half, we will load about 40 MB less of data per minute using Q-L/MRP instead of L/MRP in our distance education scenario. This will increase the available system bandwidth.

5.2.2 Dynamic Versus Static Preloading Strategies

In case of pure linear playback, the three preloading strategies, Q-L/MRP$_{\mathrm{dyn}}$, Q-L/MRP$_{\mathrm{min}}$ and Q-L/MRP$_{\mathrm{max}}$, all replace the same amount of data. However, in an interactive scenario there is a risk of prefetching too much data as the user for example might jump backwards. In this case, the

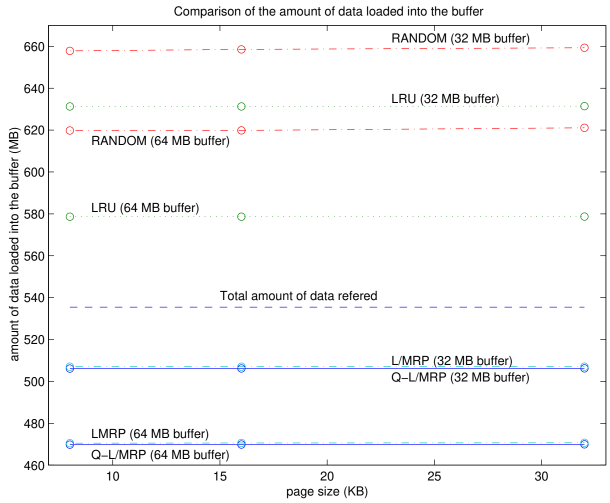

Figure 9. Page replacements for different paging strategies (interactive scenario, one user).

prefetched data will not be used, and we waste some I/O bandwidth loading data we do not need. Table 6 presents the amount of data in MBs loaded into the buffer for all the simulated preloading strategies in Q-L/MRP using the network delays given in Figure 6 and assuming data has to be retrieved from the server disk. Since Q-L/MRP$_{\mathrm{max}}$ is static and prefetches a larger amount of data than necessary in all the different network loads, the total size of preloaded data is equal using the same buffer and page size. Prefetching too much data as in Q-L/MRP$_{\mathrm{max}}$ results in loading 40 MB of data which is not used. Q-L/MRP$_{\mathrm{min}}$ always preloads a smaller amount of data which obviously results in less wasted bandwidth. Our simulations show that Q-L/MRP$_{\mathrm{dyn}}$ preloads a little more than Q-L/MRP$_{\mathrm{min}}$, which preloads from 1.2 MB to 6.2 MB more using a 32 MB buffer, but it is much better than Q-L/MRP$_{\mathrm{max}}$, which preloads from 37.2 MB to 42.1 MB less using a 32 MB buffer.

Algorithm	*Network load / Buffer size (MB)*						*Page size (KB)*
	1/32	**2/32**	**3/32**	**1/64**	**2/64**	**3/64**	
Q-L/MRP$_{\mathrm{dyn}}$	502.8	499.2	503.0	468.6	463.6	470.5	**8**
	502.2	498.2	503.0	468.3	464.1	469.1	**16**
	502.3	498.3	503.1	468.4	464.2	469.2	**32**
Q-L/MRP$_{\mathrm{max}}$	540.3	540.3	540.3	507.8	507.8	507.8	**8**
	540.3	540.3	540.3	507.8	507.8	507.8	**16**
	540.4	540.4	540.4	507.9	507.9	507.9	**32**
Q-L/MRP$_{\mathrm{min}}$	496.7	497.0	496.4	462.4	463.3	462.4	**8**
	496.7	497.0	496.8	462.5	463.7	462.7	**16**
	497.2	497.0	496.8	462.9	463.8	462.8	**32**

Table 6. Amount of data loaded into the buffer depending on preloading strategy.

When the needed data is found in the server buffer, Q-L/MRP$_{\mathrm{max}}$ prefetches the same amount of data as when data is retrieved from disk. Q-L/MRP$_{\mathrm{dyn}}$ adapts to the reduced time to retrieve data and configures like Q-L/MRP$_{\text{min}}$: less data is prefetched at a higher frequency.

5.2.3 Data Reuse Versus Buffer Size

We have looked at the gain of data reuse by increasing the buffer size from 16 MB to 128 MB. By increasing the buffer size from 16 MB to 32 MB, we get a large improvement of data reuse, and as we see in Figure 8 and 9, the amount of preloaded data decreases strongly when increasing the buffer from 32 MB to 64 MB in both single and multi-user mode. The gain of data reuse in single-user mode decreases by increasing the buffer size from 64 MB to 128 MB, because nearly all data to be presented after an interaction in our reference string can be kept in the buffer. However, in multi-user mode we can increase the buffer size to benefit more from data reuse. Compared to the amount of referenced data, Q-L/MRP in pure linear playback and multi-user mode preloads less data than referenced using a buffer of 64 MB or larger. In single-user mode (interactive playback scenario), Q-L/MRP always loads less data then the amount of referenced data.

5.3. Algorithm Execution Time

Since Q-L/MRP computes the relevance value of each COPU in the buffer according to each interaction set, it is a quite expensive algorithm to execute. The total execution time can be estimated by the function

$$\begin{aligned}\text{execution time (p,b,u,m,c)} &\approx \text{time to set relevance values + time to prefetch data}\\ &\approx \text{[\#interaction sets} \times \text{\#buffers] +}\\ &\quad \text{[\#COPUs to load} \times \text{\#pages per COPU]}\\ &\approx [(\text{u} \times \text{m}) \times \tfrac{b}{p}] + [\text{c} \times \tfrac{\text{COPU size}}{p}]\end{aligned}$$

where p is page size, b is buffer size, u is number of users, m is number of media types, and c is the number of COPUs to be preloaded into the buffer. However, all these parameters are not constants and may vary, e.g., the number of COPUs to be preloaded changes according to the workload and the COPU size varies according to each data type. Figure 10 and 11 presents some examples from our simulations[5]. The execution time increases linearly with more users and larger buffer size, and exponentially with a smaller page size. Furthermore, Figure 12 shows the gain in execution time by introducing our prefetching daemon in Q-L/MRP compared to the prefetching in L/MRP at the client side.

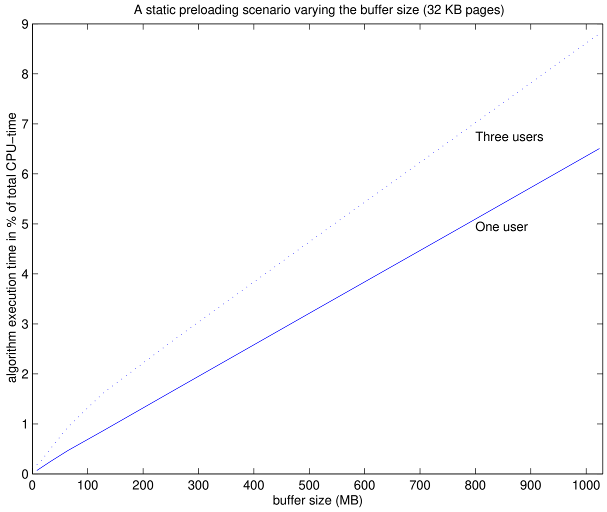

Figure 10. Average execution time of Q-L/MRP varying buffer size in a static preloading scenario.

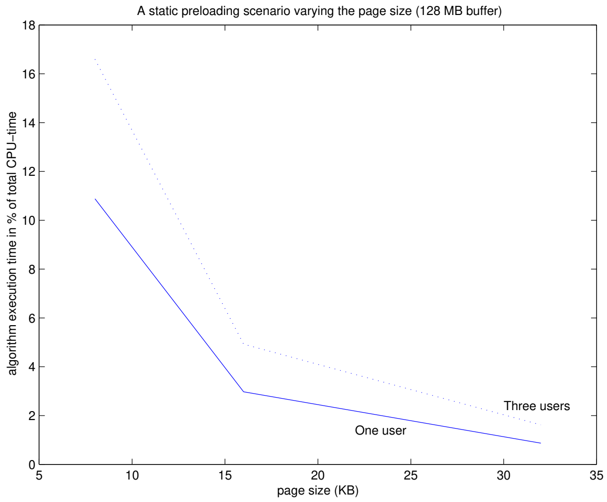

Figure 11. Average execution time of Q-L/MRP varying page size in a static preloading scenario.

6. Discussion and Conclusions

In this paper, we have demonstrated that demand paging cannot deliver data fast enough to support a continuous presentation of time-dependent data. Prefetching of data instead minimizes the number of disk accesses, and reduces the total time spent transferring data from disk to buffer.

Based on the special requirements of our DEDICATION pre-study, i.e., building a cheap prototype system for asynchronous distance education, we have designed the buffer management mechanism Q-L/MRP. Q-L/MRP is a buffer preloading and page replacement mechanism for multimedia applications with heterogeneous QoS requirements. Q-L/MRP extends L/MRP [11] with two features: (1) it supports multiple concurrent users, and (2) it supports QoS with a dynamic prefetching daemon. The dynamic prefetching daemon is able to dynamically adapt to the changes in network and disk I/O load. Furthermore, QoS requirements of users like frame rate are mapped into the buffer mechanism. Based on different interaction sets, the COPUs are assigned relevance values which are used to prefetch and replace data in the buffer.

Our performance evaluation shows that the adaption to changes in network and disk I/O load as well as the consideration of QoS requirements results in much better performance: the number of page faults is drastically reduced. For example, the higher the network delay, the better the improvements of Q-L/MRP through adaption. Prefetching of data reduces the disk I/O and network I/O bottleneck, but it contains also the danger of preloading the wrong data, i.e.,

[5] We have used the software monitor *Quantify* to calculate the number of CPU cycles needed, and computed the execution time simulating a 200 MHz CPU.

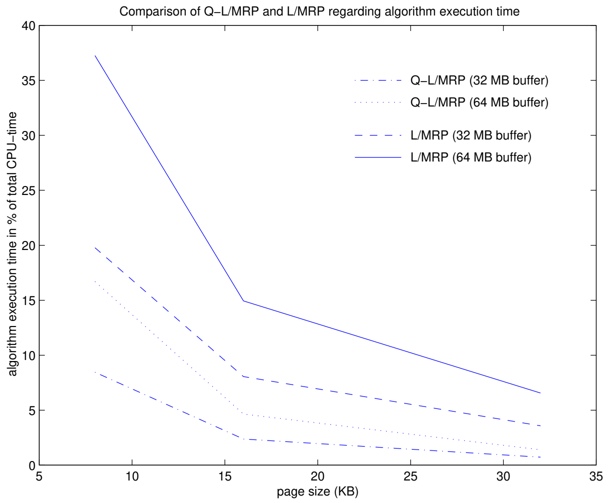

Figure 12. Execution time of Q-L/MRP and L/MRP.

data that is not used by the application. This danger is especially high for interactive usage of multimedia data. Our results show that the amount of wrongly preloaded data in Q-L/MRP is quite small, even for reference strings of interactive application usage.

In summary, we have shown that Q-L/MRP is very suitable for the special environment in our DEDICATION prestudy and outperforms other buffer management mechanisms. Since we have implemented Q-L/MRP in software only, and it benefits from the predictability of the reference behavior in continuous data streams (which is the case in many multimedia applications), the results presented in this paper do not only affect our application. They may also be valid for other applications with continuous, time-dependent data streams on other systems which may have different hardware configurations and workloads. With respect to the general applicability of Q-L/MRP in MMDBSs we detected only one restriction: the time to compute relevance values and preload data into the buffer in Q-L/MRP is increasing with a larger buffer size, more users, and a smaller page size. Thus, the Q-L/MRP algorithm might be too CPU intensive for MMDBS servers with large buffers. For MMDBS clients, however, Q-L/MRP is very suitable, as we have shown in our performance evaluation.

A direct continuation of the work presented in this paper is the analysis of the situation when using Q-L/MRP in a MMDBS client and another buffer management mechanisms, e.g., some kind of media caching [2], [7], in the MMDBS server. Generally, our ongoing and future work is concerned with QoS support from the entire system including the MMDBS for asynchronous teaching. Currently, there are two projects at UniK that work with QoS issues in MMDBSs: DEDICATION and OMODIS (Object-Oriented Modeling and Database Support in Distributed Multimedia Systems. Main emphasis in OMODIS is on the conceptual aspects and the integration of QoS in MMDBSs [4].

References

[1] Bakke, J., W., Hestnes, B., Martinsen, H.: *"Distance Education in the Electronic Classroom"*, Televerkets Forskningsinstitutt, report TF R 20/94, 1994

[2] Dan, A., Sitaram, D.: *Multimedia Caching Strategies for Heterogeneous Application and Server Environments*, Multimedia Tools and Applications, Vol. 4, No. 3, May 1997, pp. 279 - 312

[3] Effelsberg, W., Härder, T.: *"Principles of Database Buffer Management"*, ACM Transactions on Database Systems, Vol. 9, No. 4, December 1984, pp. 560 - 595

[4] Goebel, V., Plagemann, T., Berre, A.-J., Nygård, M.: *"OMODIS - Object-Oriented Modeling and Database Support for Distributed Systems"*, Norsk Informatikk Konferanse (NIK'96), Alta, Norway, November 1996, pp. 7 - 18

[5] Hollfelder, S., Kraiß, A., Rakow, T., C.: *"A Buffer-Triggered Smooth Adaption Technique for Time-Dependent Media"*, technical report No. 1002, German National Research Center for Information Technology (GMD), June 1996

[6] Johnson, T., Shasha, D.: *"2Q: A Low Overhead High Performance Buffer Management Replacement Algorithm"*, Proceedings of the 20th IEEE VLDB Conf., Santiago, Chile, 1994, pp. 439 - 450

[7] Kamath, M., Ramamritham, K., Towsley, D.: *"Continuous Media Sharing in Multimedia Database Systems"*, 4th Int. Conf. on Database Systems for Advanced Applications (DASFAA'95), Singapore, April 1995, pp. 79 - 86

[8] Liskov, B., Adya, A., Castro, M., Day, M., Ghemewat, S., Gruber, R., Maheshwari, U., Myers, A.C., Shrira, L.: *"Safe and Efficient Sharing of Persistent Objects in Thor*, SIGMOD Conference 1996, Montreal, Canada, 1996, pp. 318 - 329

[9] MATLAB (Version 4.2c), High-Performance Numeric Computation and Visualization Software, User's Guide and Reference Guide, The MATH WORKS Inc, 1992

[10] McVoy, L., Staelin, C.: *"lmbench: Portable Tools for Performance Analysis"*, USENIX 1996 Annual Technical Conf., San Diego, CA, January 1996

[11] Moser, F., Kraiß, A., Klas, W.: *"L/MRP: A Buffer Management Strategy for Interactive Continuous Data Flows in a Multimedia DBMS"*, Proceedings of the 21th IEEE VLDB Conf., Zurich, Switzerland, 1995, pp. 275 - 286

[12] Ng, R., T., Yang, J.: *"Maximizing Buffer and Disk Utilization for News-On-Demand"*, Proceedings of the 20th IEEE VLDB Conf., Santiago, Chile, 1994, pp. 451 - 462

[13] O'Neil, E., J., O'Neil, P., E., Weikum, G.: *"The LRU-K Page Replacement Algorithm For Database Disk Buffering"*, Proceedings of the 1993 ACM SIGMOD Int. Conf. on Management of Data, Washington, D.C., USA, May 1993, pp. 297 - 306

[14] Plagemann, T., Goebel, V.: *"Experiences with the Electronic Classroom: QoS Issues in an Advanced Teaching and Research Facility"*, 5th IEEE Workshop on Future Trends in Distributed Computing Systems, FTDCS'97, Tunesia, Tunis, October 1997, pp. 124 - 129

[15] Rakow, T., C., Neuhold, E., J., Löhr, M.: *"Multimedia Database Systems - The Notion and the Issues"*, Datenbanksysteme in Büro, Technik und Wissenschaft, BTW'95, GI-Fachtagung, Dresden, Germany, March 1995, Springer, 1995, pp. 1 - 19

[16] Rotem, D., Zhao, J., L.: *"Buffer Management for Video Database Systems"*, Proceedings of the 11th Int. Conf. on Data Engineering, Tapei, Taiwan, March 1995, pp. 439 - 448

[17] Seagate, http://www.seagate.com/disc/elite/elite.shtml, 1997

[18] Steinmetz, R., Nahrstedt, K.: *"Multimedia: Computing, Communications and Applications"*, Prentice Hall, 1995

[19] Sæthre, K., A.: *"Distributed Multimedia Applications and Quality-of-Service"* (in Norwegian), Master Thesis, University of Oslo, UniK, November 1996