MULTI-CLIP QUERY OPTIMIZATION IN VIDEO DATABASES

Ahmed Mostefaoui, Lionel Brunie

Information Systems Engineering Lab.
Insa de Lyon, France
Ahmed.Mostefaoui(Lionel.Brunie)@insa-lyon.fr

Harald Kosch, László Böszörményi

Institute of Information Technology
University Klagenfurt, Austria
harald.kosch(laszlo.boeszoermenyi)@itec.uni-klu.ac.at

ABSTRACT

A multi-clip query requests multiple video clips. In this paper we address the multi-clip query optimization problem. We propose a new heuristics called Restricted Search Interval that maximizes clip sharing between queries and consequently reduces the workload of the video server. The experimental results show that the suggested heuristics reduces the server workload by about 68.7% in comparison to a classical heuristic approach.

1. INTRODUCTION

Many recent multimedia applications use the multi-clip query paradigm. In such a paradigm, the result of a query is a set of continuous objects (audio or video) that need to be retrieved from a video server and delivered/displayed to the user. Some sample applications include : (a) **Customized News-On-Demand** in which a user may submit the following query : *"Show me the news clips of the day"*. The result of such a query includes all news clips (politics, sport, economics, etc.) related to that day. A user may even ask for specific news like *"show me all the highlights of the basketball matches of the weekend"*. (b) In **Video Editing** applications [1], an object is composed of a number of clips with strict temporal relations between them. Hence, delivering an object is synonymous to delivering the various clips that compose that object. (c) Many **Electronic Commerce** applications also use the multi-clip paradigm. Take for example a request made to a record company for twenty clips of the latest soul music to hit the charts. In such a case short samples (15-20 seconds each) will be delivered to the user who will make a selection.

The management of multi-clip queries poses a number of problems for the server. Firstly, there can be a number of possible delivery scenarios for a submitted multi-clip query[1]. This is due to the presentation flexibility permitted by an application. For example, a presentation of 3 clips with no ordering constraints has 6 different possible delivery scenarios. The task of the server is to find the optimal delivery scenario according to a given *optimization metrics*. Secondly, applications may specify complex presentation scenarios by imposing structural and temporal constraints on delivery scenarios (e.g. see the advanced capabilities of the JavaMediaFramework 2.0 [2] or presentation languages as those defined in [3, 4, 5, 6]). In video editing applications, for example, the clips of an object are strictly ordered, in News-On-Demand applications clips can be partially ordered, while there is generally no clip ordering in electronic commerce applications. The server must then respect this ordering when delivering clips of submitted presentations.

The constraints that can be imposed by users and applications on a presentation fall into two categories [7] : precedence constraints related to the ordering of clips when delivering them; delay constraints related to the waiting time that users/applications can tolerate. We distinguish the following delay constraints :

Max_Startup : the maximum waiting time an application can tolerate between the moment the query is submitted and the moment the first clip is delivered.

Max_Delay : the maximum waiting time between the delivery of two successive clips.

Min_Delay : the minimum waiting time imposed on the server for the delivery of two successive clips. This constraint is necessary in video editing applications, for example, where the processing of clips is rather costly in terms of computing and storage resources. It is therefore crucial for such applications not to be forced to receive successive clips too early, in which case resource problems could arise.

To illustrate the presentation optimization problem, let us consider the following example. Let us consider three presentations P_1, P_2 and P_3, such that :

$P_1 = \{(c_1, 15, 1.5), (c_2, 10, 3), (c_3, 15, 1.5), (c_4, 15, 3)\}$
$P_2 = \{(c_5, 10, 1.5), (c_6, 15, 1.5), (c_7, 10, 1.5)\}$
$P_3 = \{(c_8, 25, 1.5), (c_6, 15, 1.5), (c_4, 15, 3)\}$

Each tuple contains the clip identity, its length and its delivery rate respectively (e.g. clip c_1 has a length of l=15s and a delivery rate of r=1.5 Mb/s). For each presentation we specify the following waiting constraints :

	Max_Startup	Max_D.	Min_D.	Precedence
P_1	10s	10s	0s	$\{(c_1, c_2)\}$
P_2	20s	20s	0s	
P_3	5s	10s	0s	$\{(c_8, c_6)\}$

We suppose that the server has an available bandwidth of 3 Mb/s. Figure 1 shows an optimal schedule for the clips of the three submitted presentations. For every clip, the optimizer attributes a *start_time* at which the clip is to be delivered. Note that e.g. clip c_6 is requested simultaneously by presentations P_2 and P_3. If no constraint violation is encountered, clip c_6 can be shared between P_2 and P_3. Such sharing is called **"piggybacking"**.

This work is partially supported by France Télécom under contract CCETT N^o 96 ME 17.

[1] We will hereafter refer to a *multi-clip query* as a *presentation*.

In many multimedia applications, including those mentioned above, a subset of clips are more frequently requested ("*hot*") than the rest of the data. For instance, in News-On-Demand applications, clips from the current day are usually far more frequently requested than those from previous days. Hence, piggybacking, whenever possible, is beneficial because shared clips require no additional server resources. This increases the throughput of the server and consequently allows the latter to support more simultaneous presentations. In this paper, we concentrate on how to maximize the effect of piggybacking when scheduling presentations.

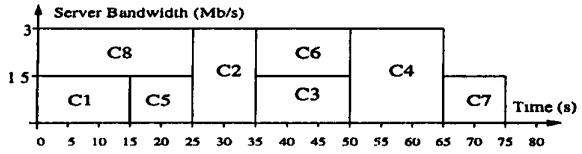

Figure 1: *Example schedule of the presentations.*

The rest of the paper is organized as follows : section 2 presents related approaches. Section 3 outlines our problem. Section 4 presents the suggested heuristics. In section 5, the effectiveness of the latter is evaluated through a series of experiments. Section 6 highlights the future work and concludes this paper.

2. RELATED WORK

To the best of our knowledge, three research works address the problem of multi-clip queries in video databases [8, 9, 10]. Shahabi et al. [8] propose an excellent formulation of the problem by defining the set of constraints that a multimedia application can impose on a presentation. However, they have not considered the potential benefit of the piggybacking. Garofalakis et al. [10] provide a near-optimal scheduling algorithm based on Graham's list-scheduling for composite multimedia objects of different length and rate. However, they did not yet include any delay constraint, nor did they considered piggybacking. Raymond and Paul [9] made the simplification that all clips in the database have the same rate and duration. They then showed that optimizing multi-clip queries is the same as finding a maximum matching in a bipartite graph.
In our work, we consider the potential benefit of piggybacking without making a reductionist hypothesis of clip size (rate and duration).

3. THE PROBLEM

The research problem that we study in this paper is *how to find* **efficiently** *an optimal or near optimal schedule of the presentation's clips that maximizes the effect of piggybacking?* In other words, given a submitted presentation with its constraints and the workload of the server (the clips of already scheduled but not yet delivered presentations), the optimizer must find an optimal schedule of the presentation's clips in a *reasonable time* such that neither the presentation constraints nor the server constraints are violated. The latter assumes that the server has enough available resources to sustain the requirements of all supported presentations. This implies that for every new submitted presentation the server checks whether or not it can be accepted (*admission control*). Deriving admission criteria for a presentation is a complex task and is highly dependent on the physical characteristics of the server (size of the buffer, disk bandwidth, striping technique used, etc.). The purpose of this paper is not to address a particular server architecture, but to propose a general framework for the multi-clip optimization problem. For this reason, we assume that the server has a maximum available bandwidth sharable among the clips (i.e. at every instant the maximum number of simultaneous delivered clips is limited). Furthermore, we assume that no overlapping can occur between the clips of the same presentation.

The task of the optimizer is then to attribute, for every clip of the submitted presentation, a start_time with respect to all defined constraints (i.e. precedence constraints, delay constraints, the no-overlapping constraint and server bandwidth constraints). The problem studied here is shown to be NP-Complete [11].

4. OPTIMIZATION ALGORITHMS

As mentioned before, the admission of a newly submitted presentation depends on the scheduling of the clips of that presentation. This implies that the scheduling task is performed on-line. Furthermore, as the server receives several presentations concurrently, the processing time taken to compute a schedule affects not only the response time of the current presentation being processed but also the response time of the other presentations. Here, we face a new constraint that is related to the computational time of a presentation schedule.

As the problem is NP-Complete, optimal algorithms have *in-facto* not been considered because of their relative long processing time. We concentrate mainly on heuristic approaches to fulfill all the constraints including the computational time of the schedule. In this section, we will present an heuristics called *Restricted Search Interval (RSI)* that attempts to maximize piggybacking. Before outlining the principle of this heuristics, we will present the basic heuristic algorithm, called the baseline heuristics, generally used to schedule the presentation clips.

Baseline heuristics The baseline scheduling heuristics is a simple and widely used list scheduling. The clips of a presentation submitted by the server are scheduled according to their order in the presentation (this order is determined by the precedence constraints). We start to schedule a clip as early as possible, i.e. for the non-first clips at the end position of the last scheduled clip plus the minimal delay which does not violate the delay and resource constraints. This approach has the advantage of being simple and fast but it does not utilize the potential benefits of piggybacking.

Restricted Search Interval Heuristics (RSI) The principle of the proposed heuristics is to merge heavy clips ($length \times delivery \times rate$ is large) already queued for schedule from previously submitted presentations. For a submitted presentation P, the heuristics operates in two main steps :

1.) Split the clips of the submitted presentation P into two clip lists, one for clips where sharing is possible (*piggybacking list*) and one for the others. Sort the piggybacking list by decreasing weight of the clips in order to maximize the effect of the piggybacking.

2.) Go through the piggybacking list and find a valid schedule for the presentation P knowing that the clip currently being examined is shared with previously submitted presentations. If there is no valid schedule for a clip in the piggybacking list (this means that piggybacking is not possible), then apply the baseline algorithm.

The second step is performed as follows : we start by considering the first element of the piggybacking list (the heaviest sharable clip). We then consider the first occurrence of the clip (less start time) in the list of already scheduled clips (submitted by previous presentations). For this occurrence we determine a *search interval*which is the maximal interval the presentation P can occupy under the given constraints. Then we try to find a valid schedule in this search interval. If this is not possible due to a constraint violation we consider the next occurrence, if any, of the clip in the list of those already scheduled. If this fails again, the next clip in the piggybacking list is considered. If no valid schedule for any of the clips in the piggybacking list can be found a timelimit is exceeded we apply the baseline algorithm.

The task of determining a valid schedule in the search interval is performed by constructing a Branch & Bound search tree. The heuristics stops when the first valid schedule is found. We illustrate the principle of the proposed heuristics in figure 2. Assume that the optimizer has to schedule the presentation P_3 of the previous example under the server workload shown in figure 2. We identify two possibilities for sharing clips c_6 and c_4. The first one is excluded because of the precedence constraint between the c_8 resp. c_6 (the corresponding search interval is not valid). Clip c_4 is shared and the corresponding Branch & Bound search tree is shown in the figure. The double edged path in the search tree corresponds to the valid schedule generated by the heuristics.

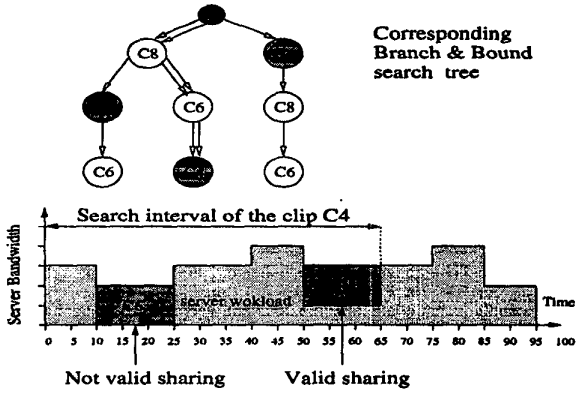

Figure 2: *Schedule of the presentation P_3 generated by the RSI heuristics.*

In order to fulfill the on-line requirement of the optimizer, we set a computational time limit for the heuristics to find a valid schedule so as not to exceed the time limit imposed. If this limit is exceeded the baseline algorithm is triggered. Note that this limit is an input parameter for the RSI heuristics and can be fixed based on the arrival rate of presentations.

5. EXPERIMENTAL ANALYSIS

This section describes a significant part of the series of experiments we performed in order to evaluate the effectiveness of the suggested heuristics. We have implemented a multi-clip query optimizer which performs admission control based on one of the following : baseline resp. RSI algorithm. Let us start by presenting the experimental settings :

- The number of presentations N_{Pres} submitted to the server was fixed at a default value of 5000. The arrival rate of presentations was modeled as a Poisson process with a mean inter-arrival time equal to 1 second. The size of the clip database varied between 50 and 750, step 50 (thus we considered 15 different values). The number of clips per presentation was chosen randomly out of an interval of 3 to 5 clips. The length of a clip was chosen randomly out of an interval of 20s to 60s. The compression rate of the clips was randomly chosen from the two compression rates MPEG-1 with 1.5 Mb/s and MPEG-2 with 4 Mb/s. The set of precedence constraints was preliminarily held empty.
- The constraints $Max\_Startup$, Max_Delay, Min_Delay, i.e. the maximal tolerable start time of the schedule, the maximal and minimal tolerable delays between two clips in a schedule of a presentation P, were set to the following typical default values : $Max\_Startup = 60s$, $Max\_Delay = 30s$ and $Min\_Delay = 0s$.

In order to evaluate the effectiveness of the proposed heuristics, we used realistic statistical distributions of clips. The first one is based on the Zipf distribution which is proven to be close to the access distribution in video rental in general and News-On-Demand in particular [12]. The second one is based on *Hot-Spot* distribution where a subset of clips (hot) are more frequently requested than others. Further variation of the parameters, delay and resource constraints are described in the report [11]. Note that for all experiments, we measured a mean computational time of **35 milliseconds** for the RSI heuristics. This computational time remains negligible in comparison to the advantage of the RSI heuristics as reported below.

Zipf Distribution In these experiments, we measured the server mean workload under the two heuristics (baseline and RSI). In our analysis we varied the zipf parameter with four distinct values : 0.9, 0.7, 0.5 and 0.3. The higher the value of the parameter, the more the same clips are requested. For a parameter value equal to 0, a uniform distribution is reached. Figure 3 displays the quotient of the server mean workload for the baseline and for the RSI heuristics. Figure 4 displays a typical server workload distribution for the baseline and for the RSI heuristics ($param_2 = 0.7$).

Figure 3 shows that a significant reduction is achieved in the server workload by applying the RSI heuristics instead of the baseline. The reduction achieved ranged from 52.7% for $param_4 = 0.3$ to 68.7% for $param_1 = 0.9$. With an increase in the size of the clip's database, the reduction decreased, but remained very significant: 14.4.% for $param_4 = 0.3$ and 39.5% for $param_1 = 0.9$.

Figure 4 gives additional details about the behavior of the two algorithms for the Zipf distribution. For larger clip databases the sharing of clips between presentations decreases. This affects the effectiveness of the heuristics.

Figure 4 shows that the server mean workload hovered around 59 MB/s with the baseline algorithm, whereas with the RSI heuristics the workload increases *fairly* sharply from 50 to 450 clips in the database and continues to increase, although *less* sharply, from 450 clips onwards. This also explains why the reduction in the server workload reported in figure 3 gradually decreases after 450 clips.

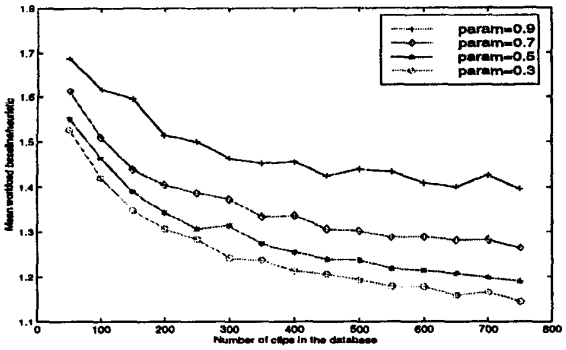

Figure 3: *Server mean workload baseline/heuristics for the Zipf distribution.*

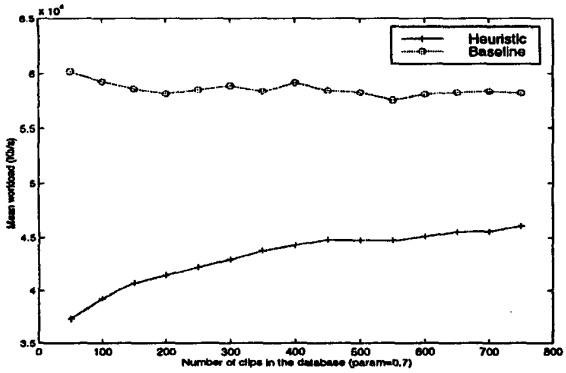

Figure 4: *Typical example of the server workload for the Zipf distribution.*

Hot-Spot Distribution In these experiments, we studied the impact of the RSI heuristics in the presence of a *Hot-Spot* distribution. We assumed that 80% of the queries (presentations) access *hot* percentage of clips in the database. We considered four values for the percentages of *hot* clips : hot_1 = 8%, hot_2 = 11%, hot_3 = 17%, hot_4 = 50%. The lower the *hot* percentage, the more the same clips are requested.

Figure 5 illustrates the quotient of the server mean workload for the baseline over the server mean workload for the RSI heuristics. The typical server mean workload distribution for the baseline and the RSI heuristics looks similar to that of the Zipf distribution (see report [11]) and the same conclusions about the scalability can be drawn.

Figure 5 shows a significant reduction of the server mean workload when using the RSI heuristics – as with the Zipf Distribution. The reduction varied between 54.2% for the case hot_4 = 50% and 95.4% for hot_1 = 8%. With an increase in the size of the clip's database, the reduction rate decreased once again, but remained very significant, 15.2% for hot_4 = 50% and 36.0% for hot_1 = 8%.

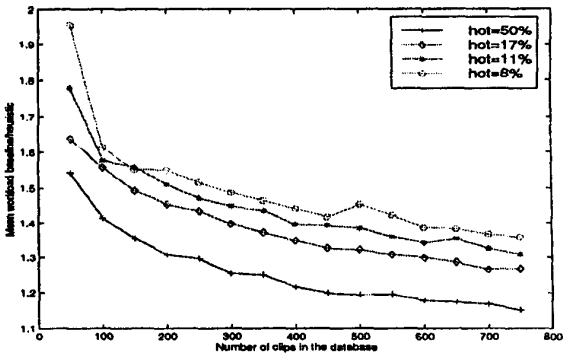

Figure 5: *Server mean workload baseline/heuristics for the Hot-Spot Distribution.*

6. CONCLUSION AND FUTURE WORK

In this paper, we tackle the problem of multi-clip optimization. We developed a novel heuristic approach that takes advantage of piggybacking. The experimental results clearly show the effectiveness of the suggested heuristics.

In future work, we plan to parallelize the proposed heuristics in order to make it faster as far as the Branch & Bound search tree gives good opportunities for parallelizing.

7. REFERENCES

[1] D.P. Anderson. Device reservation in audio/video editing systems. *ACM Transactions On Computer Systems*, 15(1):111–133, 1997.

[2] Sean C. Sullivan, Loren Winzeler, Jeannie Deagen, and Deanna Brown. *Programming with the Java Media Framework.* Wiley and Sons, Wiley Computer Books, 1998.

[3] S. Adali, M.L. Sapino, and V.S. Subrahmanian. A multimedia presentation algebra. In *ACM SIGMOD Conference*, pages 121–132. Philadephia, Pennsylvania, June 1999.

[4] N. Aloia, M. Matera, and F. Paterno. Presentations for databases in multimedia environments. *Multimedia Systems*, 6(6):408–420, 1998.

[5] C. Baral, G. Gonzalez, and A. Nandigam. Sql+d : Extended display capabilities for multimedia database queries. In *ACM International Conference on Multimedia*, pages 109–114, September 1998.

[6] S. Marcus. *Multimedia Database System : Issues and Research Direction*, chapter Querying multimedia Databases in SQL. Springer, 1996.

[7] C. Shahabi, A.I. Dashti, and S. Guandeharizadeh. Profile aware retrieval optimizer for continuous media. In *World Automation Congress (WAC)*, May 1998.

[8] C. Shahabi, A.I. Dashti, and S. Ghanderharizadeh. Continuous media retrieval optimizer and hierarchical storage structures. In *Third International Conference on Integrated Design and Process Technology IADT'98*, pages 360–367. Berlin, Germany, 1998.

[9] T. Ng. Raymond and S. Paul. Optimal clip ordering for multi-clip queries. *VLDB journal*, 7(4):239–252, December 1998.

[10] M. Garofalakis and Y. Ioannidis. Resource scheduling for composite multimedia objects. In *Proceedings of the International Conference on Very Large Databases*, pages 74–85, New York City, USA, August 1998.

[11] Ahmed Mostefaoui and Harald Kosch. Multi-clip query optimization in video servers. Technical report, Information Systems Lab (LISI), INSA de Lyon, December 1999.

[12] A. Dan, D. Sitaram, and P. Shahabuddin. Scheduling policies for an on-demand video server with batching. In *ACM International Multimedia Conference*, pages 15–23. San Francisco, October 1994.