# Scheduling Issues in Multimedia Query Optimization

MINOS N. GAROFALAKIS and YANNIS E. IOANNIDIS

*Computer Sciences Department, University of Wisconsin, Madison, Wisconsin*
*⟨{minos,yannis}@cs.wisc.edu⟩*

This paper examines the scheduling of concurrent requests on multimedia storage servers consisting of multiple resources, which is a central issue in processing and optimization of complex queries in multimedia database systems [Chaudhuri 1994]. It introduces a formal model for the demands imposed by multimedia requests on the server resources, described a heuristic algorithm for scheduling the requests that is based on the formal model, and presents a formal result that bounds the performance of the schedule produced by the algorithm compared to the optimal schedule.

Scheduling requests that involve *continuous media* data types, for example, audio and video objects, is a challenging problem due to the following distinguishing characteristics of multimedia environments:

(1) Service requests to servers are *delay-sensitive*, having specific real-time constraints for successive media quanta.

(2) Service requests to servers are typically *multidimensional*, in the sense that they alternate between multiple resources of the server during their execution.

For example, consider a content-based video-retrieval request issued from a remote client site, such as asking for sequences of digital video frames showing the Parthenon. With respect to point (1), the server must ensure that the video frames are delivered to the user at some prespecified rate to avoid "glitches." With respect to point (2), this request imposes a significant load on the server's magnetic and/or optical disk(s) that store the video object's frames, the server's CPU for identifying the frames containing the Parthenon, and the server's network interface connecting it to the remote client for transferring the selected frames.

On the basis of the architectural characteristics of the multimedia storage server and database statistics (e.g., declustering of objects across storage devices, selectivity estimates), the rate requirements of a service request can be decomposed into a vector whose components describe the bandwidth required from individual resources. This decomposition lies at the core of our mathematical framework for multidimensional resource scheduling in shared multimedia servers. More precisely, the basic premises of our modeling approach are the following:

—A media server consists of a finite collection of time-shareable resources. Each resource is characterized by an effective *service rate* (i.e., its effective

---

The work of Y. E. Ioannidis was partially supported by the NSF under grants IRI-9113736 and IRI-9157368 (PYI Award) and by grants from IBM, DEC, HP, AT & T, Informix, and Oracle.

bandwidth), which is assumed to be uniform during system operation.

- A continuous media service request consists of (a) a multidimensional *rate vector* ($\overline{R}$) whose components denote the bandwidth requirements from individual resources, and (b) a response time ($T$) indicating the amount of time for servicing the request *once it has been scheduled for execution.*

Intuitively, the components of the rate vector denote the amount of resource bandwidth that the service request must consume for the duration of its execution to ensure smooth delivery of the media strand(s) to the user. This multidimensional model of resource usage was inspired by our previous work on resource scheduling for parallel database systems [Garofalakis and Ioannides 1995].

Consider the content-based video retrieval example mentioned earlier, and assume a media server consisting of a single CPU, three magnetic disks, and a single network interface, corresponding to rate vector dimensions 1 through 5, respectively (Figure 1(a)). Also, assume that the retrieved video stream is uncompressed NTSC "network-quality" video with a consumption rate requirement of $r = 45$ megabits per second (Mbps) that has been declustered across the three disks using a round-robin policy. Finally, let $\sigma$ denote the selectivity of the retrieval (i.e., the fraction of the disk-resident frames that show the Parthenon) and $d$ the total size of the disk resident stream. Under our model, this request is described by the ordered pair $\langle \overline{R}, T \rangle = \langle [i(r/\sigma, r/3\sigma, r/3\sigma, r/3\sigma), r],\ d\sigma/r \rangle$, where $i$ is the average number of CPU instructions executed per bit processed.[1] An implicit assumption here is that the architecture of the server allows the CPU, the disk subsystem, and the network interface to operate in a fully pipelined manner (e.g., sufficient buffering). Different architectural assumptions about the server can result in different ⟨rate vector, response time⟩ pairs.

Based on the preceding model, the resource-scheduling problem for multimedia requests is abstractly defined as follows:

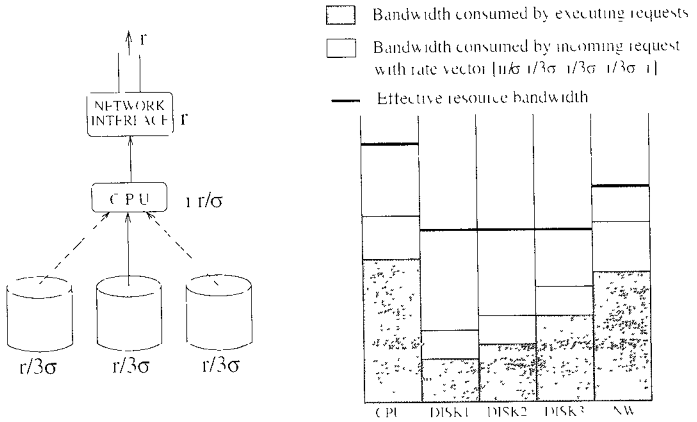
- *Given*: A collection of concurrent service requests, each with a prespecified ⟨rate vector, response time⟩.
- *Find*: A schedule of usage for the server's resources that (a) satisfies the individual bandwidth requirements of the requests without allowing the total load on any resource to exceed its availability (i.e., its effective bandwidth); and (b) optimizes some system metric, for example, schedule length, system throughput, or average response time (Figure 1(b)).

Except for trivial cases, the defined scheduling problem is clearly $\mathcal{NP}$-hard, as the multidimensional nature of continuous media access introduces a wide range of possibilities for sharing the server's resources among concurrent requests. Although the complexity of the overall query optimization process increases, one cannot afford to ignore these scheduling issues during optimization, because effective scheduling and resource sharing can substantially improve system performance [Chaudhuri 1994].

Given the intractability of the problem, we have developed a heuristic algorithm that belongs to the class of *list-scheduling* algorithms originally proposed by Graham [1969]. Our scheduler initially arranges the requests in some arbitrary list $L$ (e.g., sorted by maximum rate vector component). Whenever sufficient bandwidth becomes available, the scheduler selects the first request in $L$ which can be serviced validly without exceeding resource availability or violating precedence constraints. Given the direct relationship of our modeling framework with the models of Garey and Graham [1975] for multiprocessor scheduling under re-

---

[1] Note that $i$ depends on the nature of the service request and is used to convert the data rate to CPU bandwidth (instructions per second).



(a) (b)

**Figure 1.** Example of a service request: (a) service rate vector; (b) resource usage schedule.

source constraints, we use their results to derive the following about our list-scheduling algorithm.

THEOREM 1. *The length of the schedule (i.e., total response time) produced by our list-scheduling heuristic for a set of concurrent service requests with no precedence constraints is always within* $d + 1$ *of the optimal schedule length, where* $d$ *is the number of resources in the server.*

This result establishes the practicality of our list-scheduling algorithm for multimedia storage servers: the algorithm is very efficient, can deal with the multidimensionality of the requests, and guarantees a near-optimal response time for a set of independent concurrent service requests.

## ACKNOWLEDGMENT

We are thankful to Surajit Chaudhuri for giving us several helpful pointers to the multimedia literature.

## REFERENCES

CHAUDHURI, S. 1994. On optimization of multimedia queries. In *Proceedings of the First International Workshop on Multimedia Database Management Systems* (San Francisco, CA), 1–5.

GAREY, M. R. AND GRAHAM, R. L. 1975. Bounds for multiprocessor scheduling with resource constraints. *SIAM J. Comput. 4*, 2 (June), 187–200.

GAROFALAKIS, N. AND IOANNIDIS, Y. E. 1995 Multi-dimensional resource scheduling for parallel queries. (Submitted for publication).

GRAHAM, R. L. 1969. Bounds on multiprocessing timing anomalies. *SIAM J. Comput. 17*, 2 (March), 416–429.