A Database-Centric Approach to Enabling End-to-End QoS for Multimedia Repositories

Yicheng Tu
Computer Sciences, Purdue University
West Lafayette, IN, 47907, USA
Email: tuyc@cs.purdue.edu

1 Problem and current research

In the past few years there has been a proliferation of various multimedia applications due to greatly enhanced performance of the Internet as well as end-point servers and workstations. These multimedia services have set new requirements on larger data throughput, higher data transfer rate, and temporal and spatial constraints on how the service should be provided. The advent of Internet and the high media-content Web, both currently based on quality over-provisioning schemes, showed QoS provisioning as not being the main current focus in many non-critical areas. Due to the difference of quality parameters in user's requests to multimedia objects, QoS provisioning leads to waste of resource and thus high denial-of-service rate. Therefore, quality critical, real-time applications in areas such as military and medical services cannot be built on top of plain TCP/IP technology for obvious reasons.

A typical Network Multimedia System (NMS), generally under the support of distributed multimedia databases over wide-area networks, consists of multiple network systems and operating platforms that are highly different in topology and capacity. To achieve end-to-end control of QoS in such heterogenous, dynamic environments, all components in the NMS must cooperate to provide guaranteed real-time service. Four layers of QoS were proposed: user QoS, application QoS, system QoS and network QoS [NCN, WGS97]. The Multimedia Support Infrastructure (MSI) and Indiana Telemedicine Incubator (ITI) [1] projects currently underway at Purdue University are examples of attempts at providing user-level quality of service over collections of distributed multimedia repositories.

In spite of the fact that research in multimedia databases has covered many key issues such as data models, systems and architectures, query languages, algorithms for effective organization and retrieval of data [HE98], little attention was devoted to the aspect of the integration of QoS support at the user level.

The current resource management strategies to ensure QoS for distributed multimedia applications fall into 2 categories: reservation and adaptation. In the former approach, the multimedia applications send requests of specific QoS requirements during the period of service call establishment to ask for resources that are involved in the data processing and transfer. There are resource managers on each layer whose responsibilities are to check whether these demands can be satisfied according to resource availability and outstanding reservations. When the requests are awarded, the resource share is reserved and allocated exclusively for the use of that application and will not be deteriorated until the application stops running. In the resource adaptation approach, multimedia applications do not make reservation of the resources they need during their lifetime. Instead, filtering and scaling mechanisms are used to make the application adapt to the current available resources by changing the QoS parameters dynamically.A combination of resource reservation and adaptation [FRS00] are reported to be promising in an environment featuring high-bandwidth requests and dynamic flow changes.

[1] http://www.cs.purdue.edu/iti and http://portals.cs.purdue.edu/icds/icds_msi

2 Our approach

In this proposal we highlight the key elements of our proposed approach. The approach is motivated by query processing and optimization techniques in distributed database management systems.

We envision users such as medical professionals accessing these repositories via a simple point and click user interface ("QoP Browser"). In addition to specifying the multimedia items of interest, the user specifies a set of desired quality parameter bounds. The quality bounds could be specified explicitly or automatically generated based upon the user's profile. The user is not aware of detailed QoS parameters but rather specifies high-level qualitative attributes (e.g. "high resolution", or "CD quality audio"). Part of our research is related to the ability to translate high level user queries or actions into queries with associated QoS and security parameters, unique to each user (User Profile). Thus a QoS enabled database will search for multimedia objects that satisfy the content component of the query and at the same time can be delivered to the user with the desired level of quality.

The proposed solution relies on intermediation through several components (two of which are the *Quality Manager* and *Plan Executor*) that enable searching, locating, composing and presenting of multimedia objects with associated QoS constraints. Each aspect of the overall process involves a set of complex issues and the interaction between the components is particularly challenging. Here we present the most important ones together with envisioned design solutions. Our goals are most closely related to those of a similar project directed by Prof. Nahrstedt [NCN, LN99]. A key difference lies in our database-centric approach versus the compiler and distributed objects approach. Our approach dynamically generates multiple alternative plans that are evaluated at run time. It therefore trades greater flexibility in finding good execution plans with increased run-time cost in comparison to the approach of compiling a plan picked by an application designer. Other related efforts include [NCN, BK99, CTCL96].

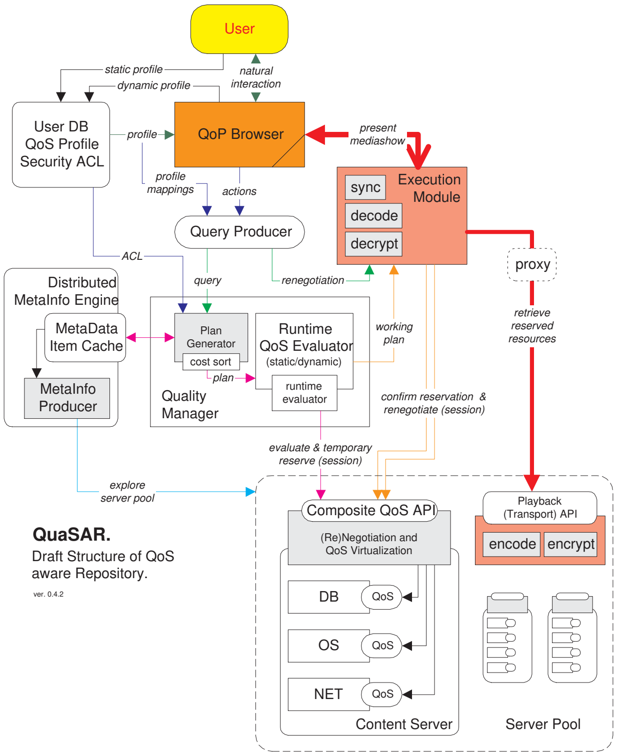

Figure 1: QuaSAR Components.

Supporting QoS in the database framework will be enabled through the creation of the following major components: a *QoP browser* front end that appends quality information to user queries; an enhanced *metadata repository* that contains media content as well as QoS related metadata; a *plan generator* that creates alternative retrieval plans based upon the user query (both in terms of the content of the data as well as the QoS constraints), it also annotates the plan with low-level constraints for each component based upon a translation of the user's QoP requirements; a *runtime QoS evaluator* that checks the feasibility of each plan under the current conditions of the system; a *composite QoS API* that interfaces to each of the underlying components to determine the current state of the system, request, and reserve resources on behalf of plans; a *transport API* and an *execution module* that together display the results to the user using the resources that have been reserved for the query plan.

Figure 1 describes in detail the proposed architecture of our QoS-enabled distributed multimedia repository which we call Quality-of-Service Aware Query Processor (QuaSAQ).

3 Research issues related to our solution

Building a distributed multimedia database system requires a careful design of many complex modules as well as effective interactions between these components. This becomes further complicated if the system is to support a non trivial aspect such as quality of service (QoS).

Important issues include: identifying and defining an acceptable set of user-level quality parameters that match the requirements of real-time distributed multimedia applications; mapping those parameters to lower-level definitions and enabling user control in their specification; the plan generation for multimedia presentations; evaluation of the costs of generated plans; provisioning the quality at all the underlying levels according to high level user specs; displaying/presenting the multimedia objects satisfying all QoS and synchronization bounds.

Resources directly involved in the QoS provisioning process are: the network, OS (CPU and memory), and storage (maybe part of the OS layer). Other components to be considered are: security (encryption etc.) provided at the server side application layer and media processing (encoding, filtering etc.) within certain quality requirements.

The key idea of our approach is to augment the query evaluation and optimization modules of a distributed database system (D-DBMS) to directly take QoS into account. To incorporate QoS control into the database, user-level QoS parameters are translated into application QoS and become an augmented component of the query. The query evaluation/optimization module takes care of resource reseravtion/adaptation to satisfy low level QoS. For this part, we are proposing the design of unified API and implementation modules, that enables negotiating with and controlling the underlying API's, virtualizing thus a single entry-point to a multitude of QoS layers (application, system, network).

Conceptually, a D-DBMS accepts a query over relations that are distributed across multiple sites and translates it into a sequence of operations to be performed at individual sites so as to compute the answer to the query. This translation is based upon metadata that describes the characteristics of the data, e.g. distribution, replication, existence of indices, and now quality. The primary criterion for identifying the best plan from among available alternatives is the cost estimate of each plan. In order to extend this D-DBMS approach to address the end-to-end QoS problem, several important requirements have to be met. These include:

- Mapping the high level user QoS parameters (e.g. resolution and window size) to low level parameters for each of the components (e.g. network bandwidth or CPU slice) has to be achieved. This may further involve negotiation between the components to arrive at an acceptable plan.
- Smart data-placement algorithms have to be developed. Copies of the same multimedia data object need to be dynamically replicated in various sites of the database. Each of these copies may have different application QoS parameters.
- A model for cost estimation is needed to evaluate the generated the QoS-aware plans. Unlike the static cost estimates in traditional D-DBMS, it is critical that the costs under current system status (e.g. based upon current load on a link) be factored into the choice of an acceptable plan. The CPU time and I/O time in the QoS setup stage are normally high therefore cannot be eliminated from the total cost formula.
- The space of possible execution plans is much larger due to the various components that must individually be considered, e.g. encryption, encoding, filtering etc. Our QoS-aware database should be able to deal with such large number of alternative plans.

- Once an acceptable quality plan has been chosen, the playback of the media objects in accordance with the required quality has to be achieved. There may be need for re-negotiating the QoS constraints due to user actions during playback.

4 Current status of the research

It is planned to start this research project by building a simplified version of QuaSAQ with dummy components upon which we can experiment on various research problems mentioned in section 3.

The implementation of QuaSAQ will be based upon the Video Database Management System (VDBMS) of Purdue University [ACE$^+$02]. Developed from the object-relational database engine PREDATOR[2] with Shore[3] as the underlying storage manager (SM), VDBMS is a DBMS that supports full-featured video operations (e.g. content-based seraching, streaming) and complex multi-feature queries. The original design of PREDATOR focused on providing the extensibility for developers to add more newly-defined data types, or Enhanced Abstract Data Types (E-ADTs) [Ses].

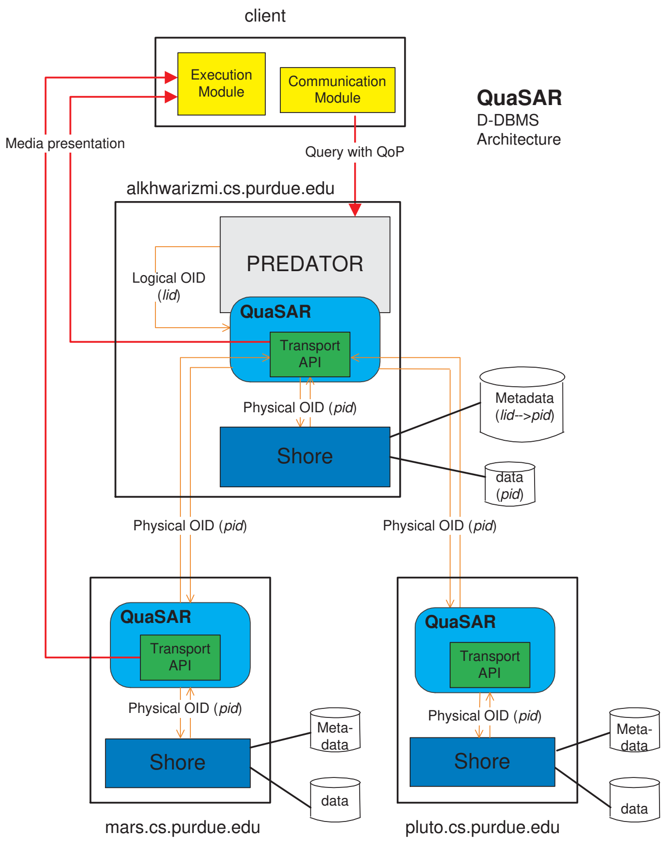

Figure 2: Experimental system architecture.

The PREDATOR code covers DBMS components from user interface to query processor and leaves data manuplation to the underlying Shore libraries. Therefore, it is Shore's responsibility to support the basic ACID requirements through two-phase locking and logging [CDF$^+$94]. Most of the VDBMS development was done by

[2] http://www.distlab.dk/predator
[3] http://www.cs.wisc.edu/shore

adding features to PREDATOR. As shown in figure 2, QuaSAQ augments PREDATOR and sits between Shore and the original PREDATOR in the query processing path. Once it gets the identity of the video object as the result of a non-QoS query from PREDATOR, QuaSAQ generates a series of plans for garaunteed delivery and chooses one to execute. It communicates with QuaSAQ modules in remote sites as well as certain Shore processes (locally or remotely, decided by the QoS plan it chooses) during plan evaluation, resource reservation, and video playback steps. Transporrt API is the module responsible for video data transmission on the server side while the Execution Module acts as data receiver on the client side . Notice the sender of the video data is not neccessarily the site at which the query was received and processed.

A number of MPEG-1 videos obtained from Indiana University Medical School will be used for this project. For each video object, 3 seed copies with different representation characteristics (high, medium, and low quality, details not shown here) are generated by a commercial software VideoMach[4]. These copies are being stored in the database and replicated in 3 individual sites running Solaris. To identify storage items (raw video, indices, relations, etc.) throughout the system, a 12-byte ID is assigned to each object according to the Shore convention.

A distributed version of Shore has been implemented using SUN RPC tools. This allows synchronozation of video retrival and manipulation. The client side was also augmented to take QoS-like SQL queries like the following:

```
SELECT vid:[s,e]
FROM video:VidLib1
WHERE (vid, s, e) IN FindVideoWithObject( QSomeoneR )
QUALITY Bitrate = High
```

5 Future Work and schedule

We have presented an overview of our database-centric approach to enabling end-to-end QoS for distributed multimedia repositories. Various issues pertaining to design and implementation have to be addressed to make the whole system work. However, my work in the following 2 - 3 months will focus on the following:

- Completion of a QoS-sensitive M-DBMS that acts as a testbed for the following research issues. The QuaSAR component in the system can be implemented as a dummy module that produces plans according to either simulation results or a predefined pattern. This may take another 2-3 weeks from now.
- Deployment of resource management inside the QoS API module. This should enable accurate estimation of contention level and controlled utilization of system resource. Another function would be to artificially generate different contention levels for the testing of system behaviour. One month has to be spent on this part.
- Evaluation of the validity of our database-centric approach to handle user requests with QoS concerning. Comparisons have to be made between our system and the original DBMS in terms of the number of video delivery requests fulfilled, as well as application level QoS parameters of these requests. This needs another month.

References

[ACE+02] W. Aref, A. C. Catlin, A. Elmagarmid, J. Fan, J. Guo, M. Hammad, I. F. Ilyas, M.S. Marzouk, S. Prabhakar, A. Rezgui, E. Terzi, Y. Tu, A. Vakali, and X.Q. Zhu. A distributed database server for continuous media. In *Proceedings of the 18th Internationl Conference on Data Engineering (ICDE '02)*, pages 490–491, February 2002.

[BK99] S. Bhatti and G. Knight. Enabling QoS adaptation decisions for Internet applications. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(7):669–692, April 1999.

[4] http://www.gromada.com/VideoMach.html

[CDF+94] M. Carey, D. DeWitt, M. Franklin, N. Hall, M. McAuliffe, J. Naughton, D. Schuh, M. Solomon, C. Tan, O. Tsatalos, S. White, and M. Zwilling. Shoring up persistent applications. In *Proceedings of ACM SIGMOD Conference on the Management of Data*, 1994.

[CTCL96] Z. Chen, S. M. Tan, R. H. Campbell, and Y. Li. Real Time Video and Audio in the World Wide Web. *World Wide Web Journal*, 1, January 1996.

[FRS00] I. Foster, A. Roy, and V. Sander. A Quality of Service Architecture that Combines Resources Reservation and Application Adaptation. In *Proceedings of the Eight International Workshop on Quality of Service (IWQOS 2000)*, pages 181–188, June 2000.

[HE98] H.Jiang and A. K. Elmagarmid. Spatial and temporal content-based access to hyper video databases. *The VLDB Journal*, 7(4):226–238, 1998.

[LN99] B. Li and K. Nahrstedt. A control-based middleware framework for quality of service adaptations. *IEEE Journal of Selected Areas in Communications, Special Issue on Service Enabling Platforms*, 1999.

[NCN] K. Nahrstedt, H. Chu, and S. Narayan. Qos-aware resource management for distributed multimedia applications. accepted to Journal on High-Speed Networking, Special Issue on Multimedia Networking.

[Ses] P. Seshadri. Relational Query Optimization with Enhanced ADTs. Technical report, Cornell University.

[WGS97] L. Wolf, C. Gridwodz, and R. Steinmetz. Multimedia Communication. In *Proceedings of the IEEE*, volume 85, pages 1915–1933, 1997.