A Multimedia Information Server with Mixed Workload Scheduling

Guido Nerjes

Department of Computer Science, University of the Saarland, Germany
E-Mail: nerjes@cs.uni-sb.de, WWW: www-dbs.cs.uni-sb.de

Abstract

In contrast to specialized video servers, advanced multimedia applications for tele-shopping, tele-teaching and news-on-demand exhibit a mixed workload with massive access to conventional, "discrete" data such as text documents, images and indexes as well as requests for "continuous data" such as video. This paper briefly describes the prototype of a multimedia information server that stores discrete and continuous data on a shared disk pool and is able to handle a mixed workload in a very efficient way.

1. Motivation

Advanced multimedia applications that make use of both discrete data (text, html, graphics) and continuous data (audio, video) include tele-shopping, tele-teaching or new-on-demand applications. In contrast to specialized video servers [1] and unlike conventional web servers, multimedia information servers have to deliver both data types to the clients. Furthermore, with unrestricted 24–hour world–wide access over the Web, multimedia information servers have to cope with a dynamically evolving workload where the fractions of continuous–data versus discrete data requests vary over time and cannot be completely predicted in advance. Thus, for a good cost/performance ratio it is mandatory that such a server operates with a shared resource pool rather than statically partitioning the available disks and memory into two pools for continuous and discrete data, respectively.

Quality of service requirements for "*continuous" data* like video and audio pose challenging performance demands on a multimedia information server. In particular, the delivery of such data from the server to its clients dictates disk–service deadlines for real–time playback at the clients. Missing a deadline may result in a temporary, but possibly user–noticeable degradation of the playback that we refer to as a *"glitch"*. Guaranteeing a specified quality of service then means to avoid glitches or to bound the glitch rate within a continuous data stream, possibly in a stochastic manner (i.e., with very high probability) [3]. In addition to the service quality guarantees for continuous data requests (*C-requests*), quality–conscious applications require that the response time of the discrete data requests (*D-requests*) stay below some user–tolerance threshold, say one or two seconds [6].

The prototype of the mixed workload multimedia information server developed at the University of the Saarland takes service quality requirements for both data types into account and implements a novel disk scheduling strategy that optimizes disk accesses. The prototype system consists of three components that are briefly described in the following sections.

2. Architecture of the Prototype System

2.1 Mixed-Workload Multimedia Information Server

The mixed workload server stores continuous and discrete data objects on a shared disk pool. A continuous data object, e.g., a video, is partitioned into *fragments* of constant time length, say 1 second of display. These fragments are then spread over the disks in a round–robin manner such that each fragment resides on a single disk. Such a coarse–grained striping scheme [2] allows a maximum number of concurrent streams for a single object (regardless of skew in the popularity of objects), while also maximizing the effective exploitation of a single disk's bandwidth (i.e., minimizing seek and rotational overhead). Furthermore, the fact that all fragments have the same time length makes it easy to support data with variable–bit–rate encoding (e.g., MPEG–2) and simplifies the disk scheduling as follows. The periodic delivery of the fragments of the ongoing data streams is organized in *rounds* whose length corresponds to the time length of the fragments. During each round, each disk must retrieve those of its fragments that are needed for a client's playback in the subsequent round. Not being able to fetch all the necessary fragments by the end of a round is what causes a glitch. On the other hand, since the ordering of the fragment requests within a round can be freely chosen, the disk scheduling employs a scan policy (also known as "elevator" or "sweep" policy) that minimizes seek times.

In contrast, a discrete data object is placed completely on a single disk. The mixed workload server schedules the service of discrete data accesses within the rounds in a way such that continuous data requests are not affected. The scheduling strategy developed to this end is a mixed, dynamic, incremental scan strategy [4,5]. This strategy mixes C–requests and D–requests within a scheduling round and is able to avoid glitches for the C–data streams by dynamically limiting the number of D–requests that are served in a scheduling round. This makes the best possible use of the remaining disk time for good response time of D–requests. Figure 1 illustrates this strategy in a simple scenario. The timeline is split into three rounds of length T. Each box shows the beginning and the end of a request execution. C-requests are shown as striped boxes, D-requests as light shaded boxes. The number in a box indicates the cylinder number where the data resides on disk. Arrows above these boxes indicate the arrival of a D-request; arrows below the boxes show that a D-request has finished and leaves the system. In each round there are two C-requests that must be served. During the first round the D-request with the number

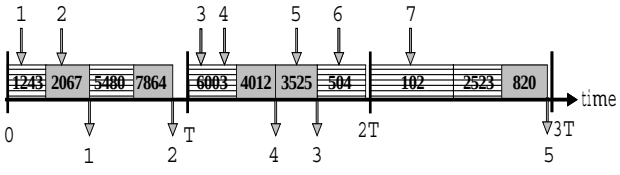

Figure 1: Dynamic and Incremental Scheduling

1 is dynamically inserted into the disk scan, since its position on disk, i.e. its cylinder number 2067, is between the cylinder numbers of the two C-requests. The global service strategy across successive rounds is fcfs, within a round it is scan. This is illustrated in the second round where two D-requests, 3 and 4, are dynamically selected and inserted into the scan. Request 4 is served ahead of request 3 because of its position on cylinder 4012. D-request 5 is delayed until the third round because there is only time for 2 discrete data requests in the second round, and request 3 must be served earlier because of the global fcfs selection between rounds.

The server is implemented in C++ with approximately 30K lines of code, and runs on Solaris and Linux. Disks are accessed using the block-based raw device interface of the operating system. The server stores meta data, e.g., object information and location of fragments on disk, permanently in files of the host file system and loads this data at startup into main memory for faster access. Measurements on a Sun Enterprise Server under Solaris have confirmed the server's scalability.

We have also developed stochastic models for predicting the performance and service quality (i.e., throughput, C-stream glitch rate, D-request response time) for a given server configuration (i.e., number of disks, etc.) [3,6]. These models drive the server's admission control for C-data streams and are also key assets towards a configuration tool for self-tuning servers.

2.2 Web Client for Content Access

The information server provides access to discrete data using the http protocol. So for discrete data access a web browser can be used that has built-in viewers for text, html and graphical documents and has the capability to launch external viewers for other media types. For video and audio playback the prototype system uses the capability of modern browsers to execute java programs. The Java Media Framework (JMF) is a package that is installed on the client and that allows to playback continuous media using java applets running in a web browser. So the simultaneous presentation of discrete and continuous data on the same web page is possible. The playback applet is stored on the information server and downloaded on-demand as a discrete data object. The functionality of the applet is to provide a user interface for the selection of video objects, to connect to the information server and to store fragments temporarily in a client buffer. The decoding of video or audio data and the display of frames is done by the JMF framework whose API is used by the playback applet. Figure 2 shows the components of the playback applet. The user is able to stop, to restart and to reposition within the continuous data stream.

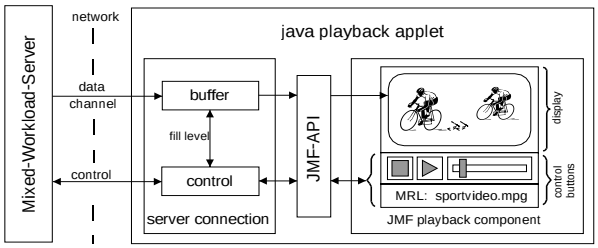

Figure 2: Components of the Java Playback Applet

2.3 Performance Monitoring and Visualization

The prototype system uses an external monitor to track and visualize the scheduling activity of the mixed workload server. It also shows the number of connected and active continuous data client. Status changes are transmitted via a permanent network connection from the server in near real-time. The graphical representation of disk accesses along the timeline on the right is similar to the representation in Figure 1. The left side of the monitor display shows the location of disk requests between the inner and the outer cylinder of the disk and the current position of the disk head. The monitor program is implemented in Java, comprises approximately three thousand lines of code, and runs in a separate process.

3. Demonstration

The demonstration shows the running prototype on two laptops connected via Ethernet. One machine hosts the mixed workload multimedia information server and the monitor program and visualizes the disk scheduling of the server. The other machine acts as a client and load generator, and presents a web browser with a sample tourist-guide application to the user. This application presents multimedia html documents with embedded text, graphics, and videos.

References

[1] Chung, S.M.: *Multimedia Information Storage and Management*, Kluwer, 1996.

[2] Özden, B.; Rastogi, R., Silberschatz, A.: *Disk Striping in Video Server Environments*, Proc. of the IEEE Intl. Conf. on Multimedia Computing and Systems, Hiroshima, June 1996.

[3] Nerjes, G.; Muth, P.; Weikum, G.: *Stochastic Service Guarantees for Continuous Data on Multi-Zone Disks*. Proc. of the 16th Symp. on Principles of Database Systems, Arizona, May 12-15, 1997.

[4] Nerjes, G.; Muth, P.; Paterakis, M.; Romboyannakis, Y.; Triantafillou, P.; Weikum, G.: *Scheduling Strategies for Mixed Workloads in Multimedia Information Servers*. Proc. of the 8th International Workshop on Research Issues in Data Engineering, Orlando, 1998.

[5] Nerjes, G.; Muth, P.; Paterakis, M.; Romboyannakis, Y.; Triantafillou, P.; Weikum, G.: *Incremental Scheduling of Mixed Workloads in Multimedia Information Servers*. To appear: Special Issue of the Journal of MM Tools and Applications, Kluwer

[6] Nerjes, G.; Muth, P.; Weikum, G.: *A Performance Model of Mixed-Workload Multimedia Information Servers*. Proc. of the 10th German Conference on Performance Evaluation, Trier, 1999.