Multimedia Performance Behavior of the GigaView Parallel Image Server

Benoit A. Gennart, Roger D. Hersch

Swiss Federal Institute of Technology (EPFL), Lausanne

Abstract

Multi-media interfaces increase the need for large image databases, supporting the capability of storing and fetching streams of data with strict synchronicity and isochronicity requirements. In order to fulfill these requirements, the GigaView parallel image server architecture relies on arrays of intelligent disk nodes, each disk node being composed of one processor and one disk. This contribution analyzes through simulation the real-time behavior of the GigaView, in terms of delay and delay jitter. For a high-end Gigaview architecture, consisting of 16 disks and T9000 transputers, we evaluate stream frame access times under various parameters such as load factors, frame size, stream throughput and synchronicity requirements.

1 Introduction

Graphic and multi-media user interfaces promote the use of computers for visualizing pixmap images. In the fields of scientific modeling, medical imaging, biology, civil engineering, cartography and graphic arts, there is an urgent need for huge storage capacities, fast access and real-time interactive visualization of pixmap images.

While processing power and memory capacity double every two years, disk bandwidth only increases by about 10% per year. Interactive real-time visualization of full color pixmap image data or compressed high-quality video streams requires a throughput of 2 to 10 Mbytes/s. Parallel input/output devices are required in order to access and manipulate image data at high speed.

A high-performance high-capacity image server must provide users located on local or public networks with a set of adequate services for immediate access to image and sound streams stored on disk arrays. The RAID concept [1, 2] offers very high bandwidth disk arrays hooked directly onto high-speed networks. The Continuous Multimedia Storage Server [3, 4] was built at Lancaster to investigate techniques required to support continuous media.

In this paper, we present a different approach : the multiprocessor multidisk approach we propose aims at associating disks and processors so as to form an array of intelligent disk nodes capable of applying parallel local preprocessing operations before sending data from the disk to the client workstation. We have shown that such preprocessing operations are highly valuable in the case of image accesses: large pixmap images can be reduced into displayable size images at disk reading speed [5]. Multimedia applications, where bandwidth must be carefully controlled, benefit from such preprocessing operations.

In the multiprocessor-multidisk (MPMD) approach [6], pixmap image data is partitioned into rectangular extents, each extent having a size which minimizes global access time. In order to ensure high throughput, image extents are stored on a parallel array of disk nodes. Each disk node includes one disk-node processor (transputer), cache memory and one disk.

The authors have implemented an MPMD image server, called the GigaView. An 8-disk T800-based architecture provides through the SCSI-2 standard interface a throughput of up to 5MBytes/s, and the ability to browse through images and maps of arbitrary size at the rate of three to four 512-by-512 3-byte-pixel visualization windows per second. Future implementations of the GigaView will rely on the faster T9000 transputer, allowing to hook up to 16 disk in parallel, and sustain athroughput of over 10MBytes/s.

This contribution analyzes through simulation the real-time behavior of the GigaView, in terms of delay, delay jitter, and buffer size requirements. For a high-end Gigaview architecture, consisting of 16 disks and T9000 transputers, we evaluate stream frame access times under various parameters such as load factors, frame size, stream throughput and synchronicity requirements.

Section 2 describes the Multi-Dimensional File System (MDFS) and the GigaView multi-processor multi-disk architecture. Section 3 analyses the behavior of the GigaView when used as a multimedia server.

Proceedings 13th IEEE Symposium on Mass Storage Systems, Annecy France, June 1994, IEEE Computer Science Society Press, 90-98

2 GigaView architecture

In this section, we describe the hardware and software architecture of the GigaView parallel image server, as well as previous research results.

2.1 GigaView Multi-Processor Multi-Disk architecture

The parallel image server consists of a server interface processor connected through a crossbar switch to an array of disk nodes (Figure 1). The server interface processor provides the network interface. Each disk node consists of a standard disk connected through a SCSI-II bus to a local processing unit. The local processors are transputers (T800 in the current versions, and T9000 when they become available). They provide both processing power and communication links. The number of links between the interface processor and the disk array is 4, equal to the number of links of a single transputer. The disk nodes support disk access, image part extraction and image reduction.

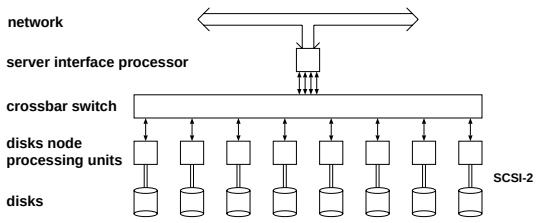

FIG. 1 : GigaView 8-disk architecture

The simulated architecture consists of T9000 transputers, rated at 50MIPS, having a memory bandwidth of 36Mbytes/s and 4 links, each transferring 8MBytes/s of data. The disks are T800-Quantum-SCSI2, with a measured 20ms seek-time and 2.23MBytes/s throughput. The seek-time is modeled as a random variable whose probability distribution is $S(t) = \frac{2}{0.06} - \frac{2}{0.0036}t$. This corresponds to a head movement between two random positions, both head positions being modeled as uniformly distributed random variables.

2.2 Multi-Dimensional File System (MDFS)

In order to access images in parallel, images are partitioned into rectangular *extents* (Figure 2). The Multi-Dimensional File System (MDFS) stores 1-dimensional (1-D), 2-D and 3-D images divided into 1-D, 2-D and 3-D extents respectively, and provides excellent access performance, regardless of the size of the accessed file and of the architecture on which it is executed [5]. Image access performances are heavily influenced by how extents are distributed onto a disk array. In a previous publication [6], we have shown that the extent size should be between 50 and 100 Kbytes, and described algorithms to allocate extents efficiently on a disk array.

0	1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16	17
18	19	20	21	22	23	24	25	26
27	28	29	30	31	32	33	34	35
36	37	38	39	40	41	42	43	44
45	46	47	48	49	50	51	52	53

FIG. 2 : Division of an image into extents

The server interface processor runs the image server master process receiving image access requests from the network and issuing image access calls to the parallel image file server. The parallel file server includes a file system master process responsible for maintaining overall parallel file system coherence (directories, file index tables, file extent access tables) and extent serving processes running on disk node processing units. Extent serving processes are responsible for serving extent access requests, for maintaining the free block lists and for managing local extent caches. Local image processing tasks required for image presentation such as image data reduction for zooming purposes are located on disk node processing units.

2.3 GigaView access modes

The *full-frame access-mode* consists of accessing all extents making up an image stored on the GigaView. This is the usual access-mode for multimedia streams.

Single image access characteristics are known : client workstations generally require rectangular portions of large pixmap image files, referred to as visualization windows. These access characteristics are referred to as the *window-selection access-mode*. In such a format, accessing a visualization window consists of fetching only the extents covered by the window. Window-selection requires two memory-copy operations : one at the disk-node processing unit to select parts of extents actually belonging to the visualization window, and the second to merge extents into one image.

2.4 Previous simulation results

Load and utilization. The load on the GigaView architecture can be expressed in absolute or relative terms. An absolute measure of the load is the average requested throughput in MBytes/s. This number gives the actual performance of the GigaView. The relative measure of the load is given in terms of system *utilization*. The utilization of a GigaView component (interface processor, links, disk nodes) is the

ratio between a the component's active-time and the total simulation time. For a set of components, the utilization of the set is the *maximum* utilization of each component. In the 16-disk T9000-based architecture of the GigaView, the disks have the maximum utilization, therefore the utilization of the GigaView is the disk utilization.

GigaView performance modeling under single requests. Simulations show that it is possible to describe the behavior of the GigaView parallel storage server using two numbers, latency and throughput. This is similar to the way secondary storage devices are described by their seek-time and throughput. The approach is to measure the delay of the parallel storage server for increasing visualization window sizes, to linearize the delay using a least-square fit, and get a formula of the type :

$$\text{AccessTime} = \text{Latency} + \frac{\text{RequestSize}}{\text{Throughput}}$$

The linearization approach has proved particularly effective, regardless of the data allocation and the architecture of the system.

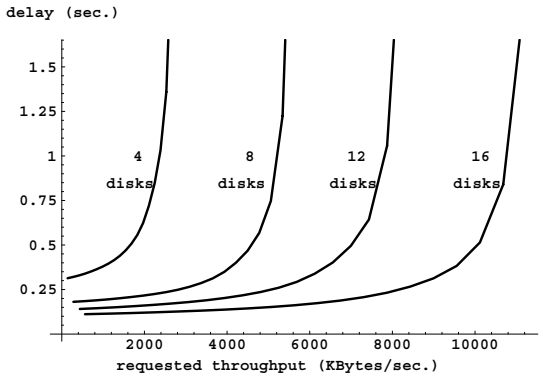

FIG. 3 : Effect of the number of disk-nodes

Multiple requests behavior. In this set of experiments, we assume that requests to the GigaView represent a Poisson process. Each experiment consists of requesting 5000 visualization windows at random positions in an image, for a given load. Each configuration is simulated for 20 loads chosen in the range of loads sustainable by the architecture. The result of each experiment is the delay average and variance over the 5000 requests. Figure 3 shows the effect of the number of disk-nodes on the performance of the GigaView. Adding disk-nodes to the architecture improves the delay of each request and the GigaView's ability to sustain higher loads. With more than 16 disks however, the interface processor bandwidth is saturated, and the performance does not increase further.

3 The GigaView as a multimedia server

This section studies the Gigaview in terms of delay and delay jitter, when its load consists of one or more multimedia streams. The purpose of the analysis of the GigaView real-time behavior is to ensure that it is possible to make it a source node in a real-time channel [7, 8]. In other words, assuming that a channel originating from or terminating at the GigaView image server has been requested and established, can the GigaView guarantee a bounded delay, for each frame in the channel, and if yes, what are the buffering requirements to guarantee the delay ? Section 3.1 describes the simulation environment. Section 3.2 and 3.3 analyze the behavior of the GigaView respectively in full-frame and window-selection access-mode. Section 3.4 analyzes the interaction between sound and image streams.

3.1 Experimental setup

During an experiment, the GigaView supplies several streams, each defined by a *request pattern*. A request pattern spans by default one second, and consists of several individual frame requests distributed over the one-second interval. To test the behavior of the GigaView under various loads, the request pattern is scaled using a factor called the *time-slice*. The one-second time-slice corresponds exactly to the request pattern described at the beginning of each experiment report. Experiments show that the utilization varies linearly with the inverse of the time-slice duration. Each experiment consists of simulating the 16-disk architecture for approximately 800 time-slices. A histogram of frame delays is gathered for each stream supplied by the GigaView, and scaled so as to represent a probability distribution. Each entry in a histogram represents a number of individual visualization window requests. To scale the histogram into a probability distribution, each entry in the histogram is divided by the total number of entries in the histogram, and again divided by the time interval covered by an entry in the histogram. A typical histogram in our simulations ranges from 0 to 400msec, with each entry representing 4ms.

All experiments consist of reading (as opposed to writing) streams. The user requests a stream from the GigaView, and the GigaView schedules each frame request. There is no jitter in the time of each frame request, since the frame requests are generated internally. In all subsequent figures, to avoid overloading the figures, only the cumulative probability distribution scale (cpd, going from 0 to 1) is shown on the y-axis.

The number of users is not specified explicitly in the experiment. When the GigaView supplies multiple streams, these streams can be isochronous and

synchronous streams requested by the same user, or independent isochronous streams requested by independent users. The number of users is only important when designing a stream scheduler, having to accommodate sets of stream requests from independent users.

3.2 Full-frame access-mode

The experiments reported on in this section describe the behavior of the GigaView in full-frame access-mode. We show results for : a single stream consisting of large frames ; three streams consisting of large frames ; four streams consisting of small frames. The results are then summarized by presenting GigaView delay curves.

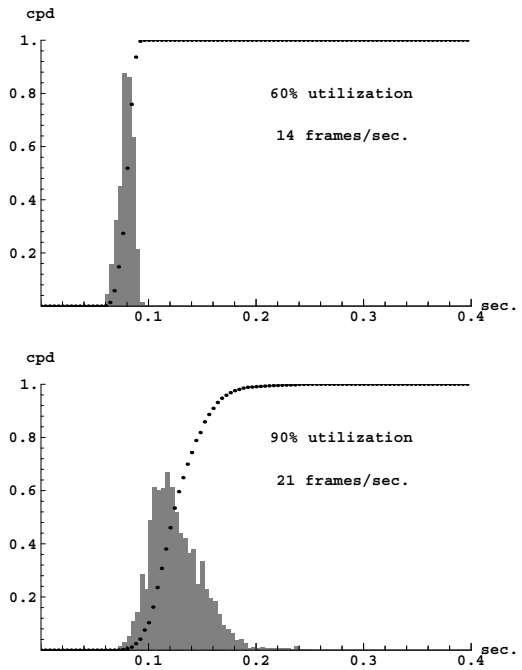

FIG. 4 : Delay jitter for single stream, large frames

Single stream, large frames. A large frame is 800KBytes in size, enough data to store a studio-quality TV image (525-by-700 2-byte pixels). It consists of 16 extents distributed on all 16-disks of the architecture. Each extent consists of 128-by-128 3-byte pixels. The one-second time-slice request pattern consists of 5 uniformly distributed frame requests. The one-second time-slice utilization is 21.45%. For an 85%-utilization, this corresponds to roughly 20 frames per second and 15.7 MBytes/s, almost the 25 frames required for standard television. Figure 4 shows the delay distribution for utilizations of 60% (358ms time-slice, 11MBytes/s, 14 frames/s) and 90% (238ms time-slice, 16.5MBytes/s, 21 frames/s). For utilizations of up to 60%, the delay jitter is roughly equal to the seek-time distribution of the simulated disks (60ms). The delay itself never exceeds 150ms for utilization of up to 80%.

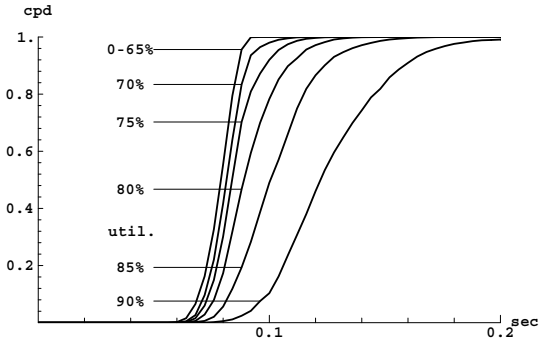

FIG. 5 : Cumulative probability distribution vs. delay and utilization, for a single stream

Figure 5 presents cumulative probability distributions (cpd) of access-delays for utilizations ranging from 10 to 90%. Each curve on the figure represents the cumulative probability distribution for a given utilization. For utilizations up to 65%, all cpd curves are virtually similar.

Multiple streams, large frames. In this experiment, the GigaView supplies three streams. The one-second time-slice request-pattern of stream one (respective two and three) consists of 5 (respective 4 and 3) uniformly distributed frame requests. The one-second time-slice utilization is 51.42%. To simulate the worst case, the three request-patterns start at exactly the same time, which causes the occurrence of three simultaneous requests for every time-slice.

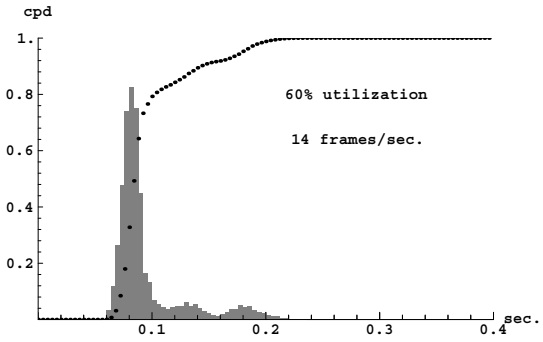

FIG. 6 : Delay jitter for multiple streams, large frames

Figure 6 shows the access delay distribution of the three combined streams. In this experiment, the GigaView utilization is 60%, or a time-slice of 857ms. The throughput is 14 frames/s, i.e. 11MBytes/s. In effect, stream interactions more than double the maximum access-delay, to 220ms, compared to the single stream access-delay performance of 100ms.

Multiple streams, small frames. 800KByte-frames are fairly large, and many multimedia applica-

tions rely on compression to reduce their data transfer rate. In this experiment, we consider 196KByte-frames consisting of 4 extents. Each extent consists of 128-by-128 3-byte pixels. The allocation of frames to disks in the architecture relies on the notion of *disk-set*. A disk-set is a subset of disks in the MPMD architecture. We consider two ways of allocating streams on the architecture : fixed disk-set, rotating disk-set. The *fixed* disk-set allocation scheme consists of allocating all frames of the same stream to the same disk-set. The *rotating* disk-set allocation scheme consists of allocating consecutive frames in the same streams on consecutive disk-sets. In this scheme, a stream is spread evenly on all disk-sets. In this experiment, the 16-disk architecture is divided into 4 disk-sets.

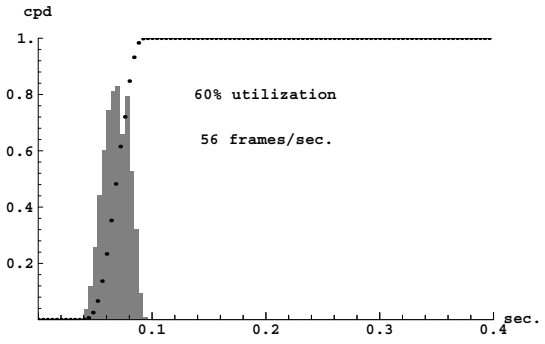

FIG. 7 : Delay jitter for small frames, fixed disk sets

Figure 7 shows the combined access delay distribution of four streams consisting of small frames, using an allocation scheme based on fixed disk-sets. Each disk-set supplies one of the four streams. All four streams consist of 11 frames/s, for a one-second time-slice load and throughput of 471ms and 8.65MBytes/s. The simulated load is 60% (785ms time-slice, 11.0 MBytes/s, 56 frames/s). Figure 7 shows that the access-delay is less than 100msec, with a jitter close to the disk jitter. There is little interaction between streams allocated on different disk sets.

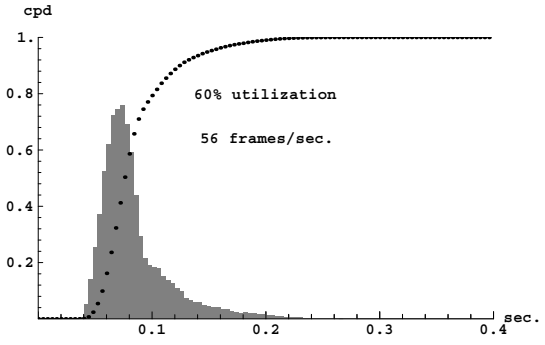

FIG. 8 : delay jitter for small frames, rotating disk sets

Figure 8 shows the access delay distribution of four streams consisting of small frames, using an allocation scheme based on rotating disk-sets. The first (respective second, third, fourth) stream consists of 11 (respective 10, 9, 8) frames per second. The load and throughput for the one-second time-slice are 40.9% and 7.47MBytes/s respectively. The simulated utilization is 60% (681ms time-slice, 11.0 MBytes/s, 56 frames/s). Figure 8 shows that the frame-access delay-jitter is much wider. Once every time slice, all four streams request a frame at exactly the same time, and one of the requests ends up having a very long delay. The maximum delay (240ms) is however much smaller than 4 times a single stream delay (100ms).

A complete comparison of the two stream allocation schemes (fixed and rotating disk-sets) requires an analysis of the number of *refused* streams. In both allocation schemes, when the utilization exceeds a certain threshold, the GigaView refuses further stream requests. In the case of rotating disk sets, the GigaView refuses a request when its *average* utilization exceeds the threshold. In the case of fixed disk-sets, the GigaView refuses a stream request when the *local* utilization exceeds the threshold. The local utilization is defined as the utilization of a single disk-set.

The assumption for this analysis is that the load requested from the GigaView consists of 4 streams, all having the same request-pattern. The 16-disk GigaView architecture is divided into 4 disk-sets. The utilization threshold is set at 90%. In the case of the rotating disk-set allocation scheme, it means that the Gigaview will refuse a request if the average load on the architecture exceeds 90%. In the case of the fixed disk-set allocation scheme, it means that a stream request will be refused if the load (i.e. the load on a specific disk-set) exceeds 90%. All 4 streams can be requested on any of the 4 disk-sets with identical probability.

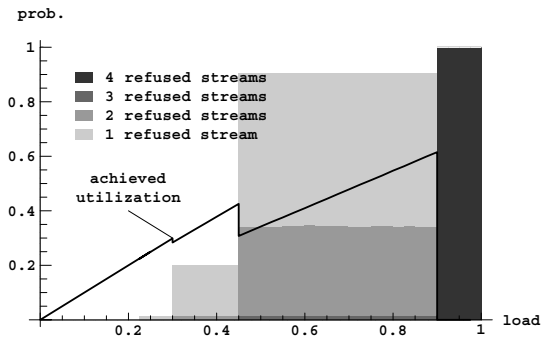

FIG. 9 : Refused streams and achieved load

Figure 9 plots, as a function of the requested utilization, (1) the probability that a stream will be refused and (2) the achieved utilization. In Figure 9, the length of a vertical cut in the white (respective light gray, medium gray, dark gray, black) part of the graph represents the probability that the GigaView will accept all four streams (respective ex-

actly three streams, exactly two streams, one stream, no stream). For example, if the 4 requested streams represent a throughput of 11MBytes/s, typical of a 60%-utilization, the probability that all 4 streams (respective exactly three streams, exactly two streams, exactly one stream) will be accepted is 9.4% (respective 56.3%, 32.8%, 1.6%). The broken line plots the achieved utilization vs. the requested utilization. The *achieved* utilization is defined as the utilization due to the accepted streams. For example, assuming 4 stream requests representing a total utilization of 60%, the achieved utilization in the case one of these requests is refused is 45%. If P_i is the probability that a stream request will be refused, N the number of requested streams, and L the utilization of a single disk-set supplying a single stream, the formula for the achieved average utilization is $\sum_{i=1}^{N} \frac{N-i}{N} P_i L$. If the requested load consists of 4 streams representing an average utilization of 60%, the achieved utilization is on the average 41%. At the average utilization of 60%, each disk set can support at most one stream. The delay jitter of each stream is therefore identical to the delay jitter presented in Figure 7.

In summary, the tradeoff between rotating and fixed disk-sets is a tradeoff between access delay and refused stream allocation. In the rotating disk-set scheme, the architecture will accept streams as long as their cumulative load does not exceed the maximum load sustainable by the architecture. The price to pay is an increased access-delay, which varies roughly linearly with the number of streams. In the fixed disk-set scheme, the architecture will maintain a very small delay (typically less than 100msec). The price to pay is in the number of refused streams.

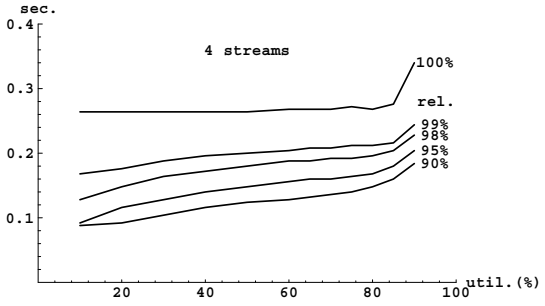

FIG. 10 : Delay vs. utilization and reliability, for small frames, rotating disk-sets

Delay vs. load and reliability, small frames, rotating disk sets. Figure 10 shows the access delay for a 16-disk GigaView supplying four streams, all consisting of small frames. The *access reliability* is defined as the probability of receiving a frame within a given delay. A 150ms delay with 95% access-reliability means that 95% of all frames will be received within 150ms. There are 5 access reliability levels, 90%, 95%, 98%, 99% and 99.99%. The utilizations actually simulated are 10%, 20%, 30%, 40%, 50%, 60%, 65%, 70%, 80%, 85%, and 90%.

Figure 10 shows that the access delay of the GigaView is relatively independent from the load for loads up to 80%, level at which the performance degrades sharply. On the other hand, the access delay is strongly dependent on the number of streams supplied by the GigaView. The streams having different request rates, frame-requests eventually collide, increasing sharply the delay of at least one of the colliding requests. For one stream, the 99.99% confidence access delay is around 100ms. For two, three and four streams, the delay increases to 160ms, 220ms and 260ms.

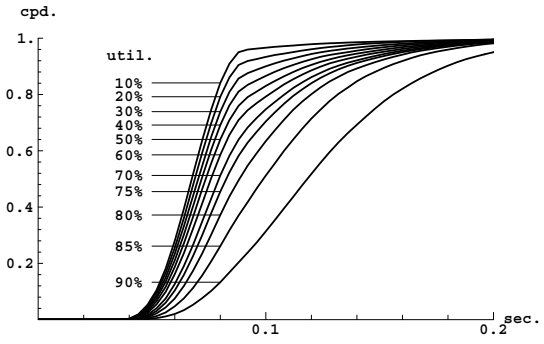

FIG. 11 : Cumulative probability distribution vs. delay and utilization, for multiple streams

Figure 11 presents the same information as Figure 10 in a different form : each curve represents the access-delay cumulative probability distribution for a given utilization.

3.3 Window-selection access-mode

Independent-request analysis [5] suggests that the GigaView architecture can effectively select visualization windows out of large images. This section analyzes the behavior of the GigaView in window-selection access-mode when supplying two image-streams. In this experiment, the frames all consist of 512-by-512 3-byte pixels (roughly 800KBytes). The images from which each frame is selected are divided in 128-by-128 3-byte pixel extents. Each frame access requires therefore 25 extent accesses. The first stream consists of 7 frames per one-second time-slice. The second stream consists of 4 frames per one-second time-slice. The utilization and throughput for the one-second time-slice is 75% and 8.65MBytes/s. The experiment whose results are reported in Figure 12 correspond to a 60%-utilization (time-slice of 1.25s, 8.8 frames/s, 6.92MBytes/s).

Figure 12 shows that the access delay of the GigaView under window selection is larger than the full-frame access-time. The effective maximum through-

put is also smaller, since in the window selection process, about 30% of the data fetched from disk is discarded. Yet the delays remain small (below 300ms), and show the ability of the architecture to accommodate multiple image streams in window-selection access-mode.

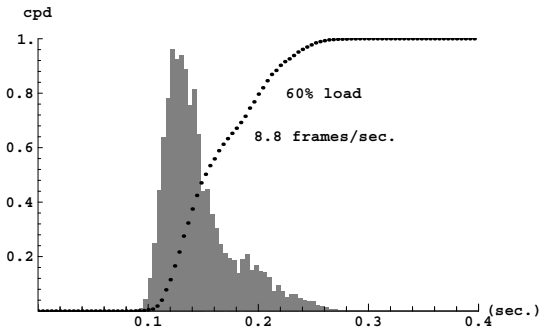

FIG. 12 : window-selection access-mode

3.4 Sound and image streams

Another issue to address is the different requirements of various multimedia streams. Loosing an image stream frame is usually acceptable. On the other hand, a late or lost sound frame is not. The ear is much more sensitive to imperfections in sound media, than the eye is sensitive to imperfections in image streams. This section analyzes two ways of allocating synchronous sound and image streams onto the disk array. It also shows that by adding a simple stream priority scheme, it is possible to keep the sound-stream frame access-delay very low.

An image stream consists of several megabytes of data per second whereas a sound stream consists only of several tens of kilobytes of data per second. We also have to take into account that the best extent size is around 50KBytes. For these reasons, we will consider in this study a one second request-pattern consisting of one sound frame (one 50KByte-extent), and ten large image frames per second (each frame is 800KBytes). The one-second time-slice utilization and throughput are 42.98% and 7.91MBytes/s.

The criteria for evaluating an allocation scheme are : balanced disk utilization, or *time-efficiency* ; balanced disk occupation or *space-efficiency* ; minimum access-delay. We address the problem of allocating frames making up one time-slice (10 image frames, and 1 sound frame).

A first way of allocating the two synchronous streams on a 16-disk architecture consists of dividing each image frame in 15, and allocate the image stream on disks 0 to 14. The sound stream is allocated on disk 15. This means that the size of the data on disk-15 will be only one tenth of the size of the data on disk 0 to 14. Assuming all disks have the same size, the disk containing the sound stream will be under-utilized, that is, the space efficiency is not optimal. The time efficiency will not be optimal either. Indeed, disk-15 utilization will be only one tenth of disk 0 to 14 utilization, since only one sound frame is accessed for every ten image frames.

A better solution consists of dividing one in every ten image frames into 15 extents, and allocate it on disk 0 to 14. The sound frame is allocated on disk 15. All other image frames are divided in 16, and allocated on all 16-disks. The sound and image frames are accessed through independent requests. This allows to give priority to the sound stream, and therefore improve access-delay of each sound frame. This allocation scheme is referred to as the *hybrid* allocation strategy. It is time- and space-efficient : during one time-slice, all disks are accessed 10 times, and all disk accesses have approximately the same size.

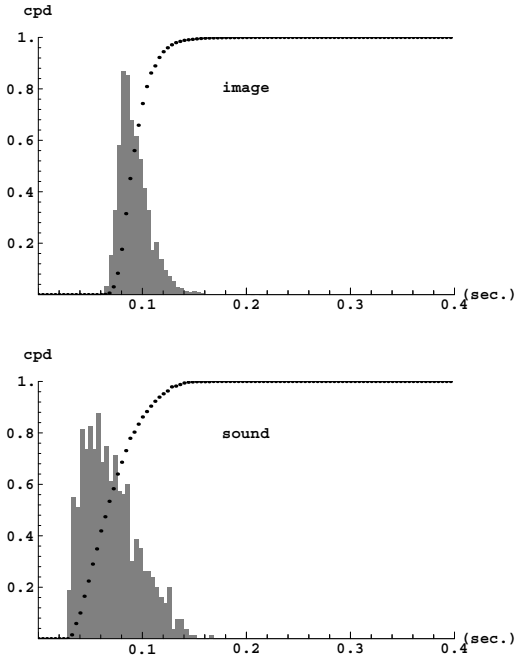

FIG. 13 : Delay jitter for image and sound streams at 80% load

It is possible to improve the sound delay and jitter by giving priority to the sound stream. Internally, the GigaView splits a frame request into extent requests. The idea is to give sound-extent requests priority over image-extent requests. Giving priority to a request means that a pending request with higher priority will be executed first. It does not mean that a request will be *preempted*, i.e. will be interrupted and replaced by a higher priority request. Indeed, interrupting an extent request to the disk will require an extra seek operation, and therefore much degrade the performance of the preempted request. Priority ensures that the sound extent request will be executed as soon as the current extent request is completed. It

improves the access-delay, especially when the load is high.

This experiment analyzes the delay and delay jitter for combined image and sound streams distributed on the disk array using the hybrid allocation strategy. The load for the experiment is 80% (537ms time-slice, 14.7MBytes/s, 18.6 image frames/s, and 1.86 sound frame/s). Figure 13 shows that indeed the sound frame delay remains well under twice the extent access-delay. Other experiments for utilizations as high as 90% confirm that the sound frame delay never exceeds 160ms.

4 Conclusion

The multimedia performance analysis of the GigaView has shown that a 16-disk T9000-based MPMD architecture is able to sustain a throughput of 15MBytes/s in full-frame access-mode, with a worst case delay inferior to 250ms, and a throughput of 9MBytes/s in window-selection access-mode, with a worst case delay inferior to 350ms. It is possible to allocate synchronous sound and image streams both time- and space-efficiently, and maintain the delay of sound frames under 160ms. All experiments highlight the importance of being able to control individual extent allocation. The stream allocation scheme based on rotating disk-sets maximizes the average load at the expense of delays, whereas the stream allocation scheme based on fixed-disk sets minimizes access delay at the expense of the average load.

References

[1] E. K. Lee, P. M. Chen, J. H. Hartman, A. L. Drapeau, E. L. Miller, R. H. Katz, G. A. Gibson, and D. A. Patterson. RAID-II : a scalable storage architecture for high-bandwidth network file service. Technical Report 92/672, Computer Science Department, University of California at Berkeley, 1992.

[2] R. H. Katz, P. M. Chen, A. L. Drapeau, E. K. Lee, K. Lutz, E. L. Miller, S. Seshan, and D. A. Patterson. RAID-II : design and implementation of a large scale disk array controller. In *VLSI System Design Conference*, Seattle, WA, March 1993.

[3] G. Blair, A. Campbell, G. Coulson, F. Garcia, D. Hutchinson, A. Scott, and D. Shepherd. A network interface unit to support continuous media. *IEEE Journal on Selected Areas in Communications*, 11(2):264–275, February 1993. Special issue on network interfaces.

[4] P. Lougher and D. Shepherd. The design of a storage server for continuous media. *The Computer Journal*, 36(1):32–42, February 93.

[5] R. D. Hersch, B. Krummenacher, and L. Landron. Parallel pixmap image storage and retrieval. In Grebe et al., editor, *Proceedings of the World Transputer Congress*, pages 691–699. IOS Press, 1993.

[6] R. D. Hersch. Parallel storage and retrieval of pixmap images. In *Proceedings of the 12th IEEE Symposium on Mass Storage System*, pages 221–226, Monterey, 1993.

[7] D. Ferrari and D. C. Verma. A scheme for real-time channel establishment in wide-area networks. *IEEE Journal on Selected Areas in Communications*, 8(3):368–379, April 1990.

[8] Domenico Ferrari. *Design and applications of delay jitter control scheme for packet switching networks*, volume 614 of *Lecture notes in computer science*, pages 72–83. Springer-Verlag, 1992.