Comparing Multimedia Storage Architectures

Benoit A. Gennart, Roger D. Hersch

Swiss Federal Institute of Technology (EPFL), 1015 Lausanne, Switzerland
{gennart,hersch}@di.epfl.ch

Abstract

Multimedia interfaces increase the need for large image databases, capable of storing and reading streams of data with strict synchronicity and isochronicity requirements. In order to fulfill these requirements, we use a parallel image server architecture which relies on arrays of intelligent disk nodes, each disk node being composed of one processor and one or more disks. This contribution analyzes through simulation the real-time behavior of two multiprocessor multi-disk architectures : the GigaView and the Unix workstation cluster. The GigaView incorporates point-to-point communication between processing units and the workstation cluster supports communication through a shared bus-and-memory architecture. For a standard multimedia server architecture, consisting of 8 disks and 4 disk-node processors, we evaluate stream frame access times under various parameters such as load factors, frame size, stream throughput and synchronicity requirements. We compare the behavior of the GigaView and the workstation cluster in terms of delay and delay jitter.

1 Introduction

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, video and sound streams stored on disk arrays. The RAID concept [1] offers very high bandwidth disk arrays hooked directly onto high-speed networks. The multiprocessor multidisk (MPMD) approach we use associates disks and processors so as to form an array of intelligent disk nodes capable of applying in parallel local preprocessing operations before sending data from the disks 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 [2]. Multimedia applications, where bandwidth must be carefully controlled, benefit from such preprocessing capabilities. In the MPMD approach, pixmap image data is partitioned into rectangular extents, each extent having a size which minimizes global access time. In order to ensure high throughput, contiguous image extents are allocated on different disk nodes. The Multi-Dimensional File System (MDFS) developed at EPFL [3] handles data partitioning and allocation on multiple disk nodes.

The authors have implemented an MPMD image server, called the GigaView. A 4-disk T800-transputer-based architecture connected through a SCSI-2 standard interface to a host computer (MacIntosh, Unix Workstation) provides 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, which can support up to 16 disks hooked in parallel, and sustain a throughput of approximately 15MBytes/s.

This contribution analyzes through simulation the real-time behavior of the GigaView, in terms of throughput and delay jitter. It compares the performance of the GigaView to the performance of a general-purpose multi-processor multi-SCSI-channel high-end UNIX workstation cluster. For high-end image server architectures, consisting of 8 disks and 4 processors, we evaluate stream frame access times under various parameters such as load factors, frame size, stream throughput and synchronicity requirements. In this contribution, we consider reading multimedia streams stored on disk, without any processing operation such as compression, decompression or resampling. This allows us to highlight the overhead due to data transfers within the image server architectures. Future contributions will take into account operations that can be executed in parallel.

Our approach is to evaluate through experiments on single-processor single-disk workstations individual component performance (e. g. local processor memory to global memory bandwidth, local processor memory bandwidth, disk throughput and latency), and use the performance parameters to build simulation models of MPMD architectures. We then evaluate through simulation the performance of the modeled MPMD architectures.

The result of the analysis is that, despite lower individual component performance, the point-to-point communication scheme supports higher throughputs at the application level and scales better to higher performance architectures.

Section 2 describes the Multi-Dimensional File System (MDFS), the GigaView multi-processor multidisk architecture, and the architecture of a multiprocessor multi-SCSI-channel workstation cluster. Section 3 discusses the models used to simulate the architectures, as well as the methodology used to evaluate each architecture parameters. Section 4 compares the throughputs of both architectures. Section 5 analyses the behavior of the GigaView and workstation

Proc. Int'l Conf. on Multimedia Computing and Systems IEEE Press, Washington, USA, 1995, 323-329

cluster when used as multimedia servers.

2 GigaView and workstation cluster

In this section, we describe the hardware and software architecture of both the GigaView parallel image server and a generic workstation cluster architecture (section 2.2). When statements apply to both the GigaView and workstation cluster architectures, we refer to them under the name : the *parallel image server*. We introduce the concepts underlying the Multi-Dimensional File System which specially supports imaging applications (section 2.3).

2.1 GigaView architecture

The Gigaview consists of a server interface processor connected through communication links to an array of intelligent disk nodes (Figure 1). The server interface processor provides the network interface. Each disk node consists of one or more standard disk(s) connected through a SCSI-II bus to a local disk node processor. The local processors are transputers (T800 in the current version, and T9000 when they become available). They provide both processing power and communication links. The number of links per transputer is 4. Data transfers through the links and data processing by the transputer do not interfere : data packets transferred through links are written by DMA (direct memory access) into the processor's memory. The disk nodes support disk access, extent caching, image part extraction and image (de)compression. Since the transputer supports context switches in hardware, contexts switches can be executed in a few microseconds, and therefore do not add any noticeable overhead to the main computations.

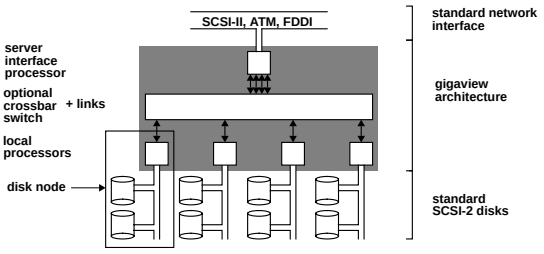

Figure 1 : GigaView 8-disk architecture

2.2 Workstation cluster architecture

The workstation cluster architecture (figure 2) consists of a single high-speed backplane bus connected to processors, SCSI-channels and main memory. The SCSI-channels connect secondary storage devices (typically, magnetic disks) to the backplane-bus. We assume that it is possible to transfer data directly from secondary storage to main memory by DMA.

2.3 Multi-Dimensional File System

In order to access disks in parallel, images are partitioned into rectangular *extents*. 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. Image access performances are heavily influenced by how extents are distributed onto a disk array. In a previous publication [3], we have shown that the extent size should be between 12 and 48 KBytes, and described algorithms to allocate extents efficiently on a disk array.

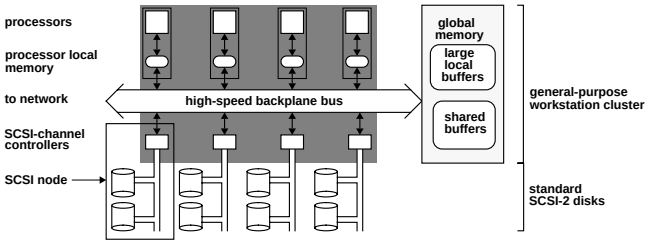

Figure 2 : Workstation cluster architecture

3 Architecture Modeling

This section describes the methodology used to model the architectures. Individual components such as memory, disks, busses, processors are measured experimentally, and relevant parameters such a throughput and latency are evaluated. Simulation models for individual component operations (e. g. the time to transfer a data packet from disk to shared memory) are created using the measured parameters. A system is modeled as a set of individual components. Operations on a system (e. g. the GigaView reading a visualization window) are specified as a series of individual component operations. Measured systems are actual systems such as a 4-disk GigaView or a single-disk single-processor workstation. Simulated systems are prospective systems such as a 4-disk-node 16-disk GigaView or a 4-processor 8-disk workstation cluster. The simulator derives the system performance for specific stimuli. The benefit of this approach is the ability to evaluate accurately architectures consisting of many processors and disks, having varying individual component performance. It allows asking questions such as : how does the processor performance affect the overall system performance ; what is the architecture bottleneck ; what is required from a specific individual component to reduce the bottleneck.

Section 3.1 describes a methodology to measure the actual performance of a system's individual component. Section 3.2 specifies the GigaView and workstation cluster simulation models.

3.1 Evaluating individual components

Experience shows that for multimedia applications (consisting essentially of data transfers), all individual components (shared memory, local memory, transputer links) exhibit a linear behavior. That is, their delay depends linearly on the data set size. Therefore, two parameters, *latency* and *throughput* are sufficient to model their behavior using the formula Delay = Latency + $\frac{\text{DataSetSize}}{\text{Throughput}}$. To evaluate throughput and latency of a given operation, we plot its delay as a function of the data set size, and linearize (least-square fit).

The slope of the linearized curve gives the throughput. The intersection with the ($DataSetSize = 0$) vertical axis gives a measure of the latency.

Definitions. We consider two software concepts : *process* and *buffer* ; and two hardware concepts : *processor* and *memory*. In the following discussion, the word *global* applies to memory accessible by all processors in an architecture ; the word *shared* applies to a buffer visible by all processes in a program ; and the word *local* is applied to the memory (resp. a buffer) visible by a single processor (resp. process). The assumptions are that (1) a small local buffer fits in the local processor memory ; (2) a large local buffer exceeds the local memory size, and is therefore stored in global memory ; (3) a shared buffer is always in global memory. These assumptions aim at producing a simple model of the general memory access behavior of a workstation cluster, where the hierarchy of caches of a processor is modeled as local memory, and global memory operations (set, copy) are modeled as a number of backplane bus transfers. The test programs enable us to confirm or invalidate these assumptions, and model the number of backplane bus transfers required by a given global memory operation.

Goal. In the multimedia application considered for this contribution, all operations consist of data transfers : reading from disks ; transferring data through the backplane bus to and from main memory ; transferring data through transputer links ; copying data in local memory. In previous contributions, we have measured disk transfers and transputer link transfer rates. The disks are rated at 10ms latency and 4MBytes/sec. The T9000 links are rated at 8MBytes/sec and 5μs latency. Our purpose is to measure the backplane bus throughput and the local memory throughput of workstation clusters.

To evaluate these two parameters, the authors wrote 7 test functions and deduced from the so obtained delay measures the performance parameters. The 6 functions are : (a) ISB, initialize a shared buffer using the UNIX memset function ; (b) IPCISB, initialize a shared buffer (memcpy) allocated using the IPC mechanism ; (c) ISLB, initialize small local buffer (memset) ; (d) ILLB, initialize large local buffer (memset) ; (e) CSLTSB, copy small local buffer to shared buffer (memcpy) ; (f) CLLTSB, copy large local buffer to shared buffer (memcpy) ; (g) SSLTSB, shuffle small local buffer to shared buffer (memcpy). Our assumption is that a memset (resp. memcpy) operation requires one (resp. two) bus transfer. The typical small buffer size is 8KBytes to 64KBytes, small enough that no data is transferred onto the backplane bus. The size of a large buffer is 1 to 8MBytes. Test functions are called repeatedly so that the typical experiment lasts about 1 sec. Our assumption is that a large-buffer memset operation corresponds to one backplane-bus data-transfer, and the large-buffer memcpy corresponds to two backplane-bus data-transfers.

To evaluate the performance parameters of various architectures, we ran the 7 test functions on single-processor workstations. Table 1 summarizes the results for 4 UNIX platforms : SparcLX station (SLX), SparcServer 1000 (S1000), Silicon Iris (Iris), and Dec Station 3400 (DEC). These are affordable workstations, with a price in the $20K range. For reference, we give the performance results of the Silicon Challenge (Chall.) with one processor.

Table 1 shows the performance for a single-processor workstation running a single process. The numbers in the table represent data transfer rates between different parts of the architecture (MBytes/s). The numbers in the table are accurate within 20%. That is, when reproducing the experiment, we get a deviation in throughput numbers that stays within 10% of the values displayed in the table.

Assuming that the ILLB (initialize large local buffer) routine measures the backplane bus throughput, and that the ISLB (initialize small local buffer) routine measures the local memory throughput, table 1 suggests that the Sparc station LX and the Sparc Server have approximately the same bus performance (33.7 and 26.5MBytes/s), and that the SPARC server 1000 has faster local memory throughput. The Silicon IRIS has a faster bus and higher local memory throughput than both Sparc architectures. The Dec Station has outstanding local memory and bus throughputs.

arch.	SLX	S1000	Iris	Dec	Chall. (1 proc.)
ISB	32.6	24.9	39.7	84.5	118.
IPCISB	9.5	13.7	28.8	55.6	118.
ISLB	32.0	52.3	74.2	294.	945.
ILLB	33.7	26.5	39.6	76.0	100.
CLLTSB	15.2	15.1	45.9	42.0	56.
CSLTSB	16.3	18.7	28.7	56.8	104.
SSLTSB	5.12	9.00	16.4	21.3	48.5

Table 1 : Workstation cluster throughput (MB/s) (single processor, single process)

Comparing the ILLB (init large local buffer, modeled as one backplane bus transfer) and the CLLTSB (large local to shared buffer, modeled as two backplane bus transfers) routines, we notice that indeed the CLLTSB throughput is roughly half the ILLB throughput, except for the Silicon IRIS. This suggests that indeed the one- and two-backplane-transfer assumptions are valid for the Sparc and Dec architectures. In the Iris architecture, a DMA mechanism may provide direct memory to memory transfers. The SSLTGB test function (shuffle small local buffer to shared buffer) shows that the cost of transferring the data in small packets is high : compared with the single packet transfer rate (copy small local buffer to shared buffer), the throughput is divided by at least a factor of 2. Comparing the ISB and IPCISB functions, we notice that the overhead due to the IPC mechanism is at least a 1.5 factor. The authors are aware that there are other mechanisms than IPC to share memory between processes, but the fact remains that there is always an

overhead for shared memory access.

For our simulations, we assume that the backplane bus throughput (resp. local memory bandwidth) is equal to the ISB (resp. CLLTSB) function throughput. We round up the numbers of table 1, and make use the numbers of table 2. Since at the time of publication, the T9000 was not yet available, we assume its performance to be 4 times the T800 performance.

arch.	T9000	S1000	Iris	Dec
backplane		30	40	75
memory	72	50	75	280

Table 2 : Workstation cluster throughput (MB/s)

3.2 Simulation models

Using the parameters measured on single-processor workstations in section 3.1, we specify models of two multiprocessor multidisk architectures : the GigaView architecture using point to point communication between processors and disk-nodes, and the workstation architecture using a shared-memory-and-bus architecture for communication.

Reading a visualization window from the GigaView consists of decomposing a window request into extent requests. As soon as an extent request is generated by the interface processor, it is transferred down the appropriate transputer link to the disk node where the extent is located. The disk-node reads the extent from the disk into its processing unit memory. The extent is then transferred up a transputer link back to the interface processor, where it is merged with the other extents to form the visualization window. For all experiments, the GigaView model consists of T9000 transputers (local memory throughput of 72MBytes/s, link throughput of 8MBytes/s).

Reading a visualization window from the workstation cluster consists of decomposing a window request into extent requests. The decomposition is carried out by one of the workstation cluster processors. As soon as an extent request is generated by the processor, it is transferred down the backplane bus to the SCSI node where the extent is located. The extent is read from the disk and transferred by direct memory access to global memory. The processor then merges the extent scanline by scanline into the visualization window located in global memory. This last operation requires two additional transfers on the backplane bus. The last bus transfer suffers from two overheads : access to shared memory and small-packet transfer. We assume that the small-packet transfer overhead to be a factor of 2 (the ratio between the throughputs of the CLLTSB and SSLTSB functions), and the overhead of shared memory access to be an additional factor of 1.5. The workstation cluster model is based on the DEC performance parameters (backplane bus throughput of 75MBytes/s, local memory throughput of 280MBytes/s).

The same disks are used in both architectures. The disks are 1GByte IBM disks rated at 10msec seek-time and 4MBytes/s throughput. The following section analyzes the GigaView and workstation cluster behavior when used as multimedia servers.

4 Architecture throughput

Simulations show that it is possible to describe the behavior of a parallel storage server using two numbers, latency and throughput. Figure 3 shows how the throughput evolves as disks are added to each architecture. The two architectures all have 4 disk-nodes. Disk-nodes consist of one processor with 1, 2, 3 or 4 disks.

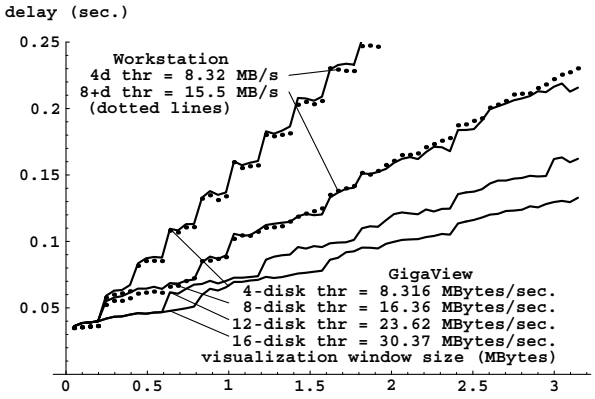

Figure 3 : GigaView vs. Workstation throughput

For the T9000-based architecture, the disks are the busiest components for up to 12 disks in the architecture. For a 12-disk architecture, the disk (resp. links, local processor, interface processor) utilization for a 3.2MBytes visualization window request is 71% (resp. 17%, 49%, 68%). Above 12 disks, the server interface processor is more utilized than the disks. In the case of the workstation architecture, the curves are superimposed for all architectures with more than 8 disks (figure 3, dotted lines). This indicates that the performance is limited not by the disk throughput but by another component. Analysis of the utilization data indicates that the backplane bus is indeed the bottleneck. With a bus rated at 75MBytes/s the application throughput is limited at 15MBytes/s, or a fifth of the backplane bus throughput.

5 Multimedia servers

This section studies both the GigaView and the workstation cluster in terms of delay and delay jitter, when their load consists of one or more multimedia streams. The purpose of the analysis is to ensure that it is possible to make both architectures a source node in a real-time channel [4]. In other words, assuming that a channel originating from or terminating at the parallel image server has been requested and established, we try to establish whether a parallel architecture can guarantee a bounded delay for each frame in the channel.

Section 5.1 describes the experimental setup. Section 5.2 analyzes the image servers' behavior for a single user reading frames allocated on multiple disks. Section 5.3 analyzes the image servers' behavior for

multiple users reading frames allocated on multiple disks.

5.1 Experimental setup

During an experiment, the parallel image server supplies one or more streams, each defined by a *request pattern*. By default, a request pattern spans one second and consists of several individual frame requests distributed over the one-second interval. To test the behavior of the parallel image servers 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 8-disk architecture for approximately 1000 time-slices. A histogram of frame delays is gathered for each stream supplied by the parallel image server and scaled so as to represent a probability distribution.

All experiments consist of reading (as opposed to writing) streams. The user requests a stream from the image server, and the image server schedules each frame request. There is no jitter in the time of each frame request, since the frame requests are generated internally. The results are presented in terms of delay probability distribution (pd) and delay cumulative probability distribution (cpd). In figures where both the delay probability distribution and the delay cumulative probability distribution are shown, only the cumulative probability distribution scale (cpd, going from 0 to 1) is shown on the y-axis.

In this set of experiments, both architectures consist of 4 storage nodes, each storage node including two disks. We compare a T9000-based GigaView architecture and a DEC-based workstation cluster architecture. The experiments reported in this section describe the behavior of the image servers in uncompressed full-frame access-mode. 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. Frames in a stream are 400KBytes in size. For reference, a studio-quality TV single-frame image consists of 720-by-525 2-byte pixels, or 756KBytes. Each frame is segmented into 8 extents distributed on all 8-disks of the architecture. We show results for single and multiple users requesting streams of frames distributed over multiple disks.

5.2 Single user

In this experiment, the image server supplies one stream. The one-second time-slice request pattern consists of 8 uniformly distributed frame requests. The following two paragraphs compare the GigaView and workstation cluster architecture, for a 32 frames/s load, corresponding to a 250ms time-slice.

Access delays. For the Gigaview (resp. workstation cluster), 32 frames/s corresponds to a 70% (resp. 85%) utilization. The shaded areas in figure 4 represent probability distribution, and the continuous lines cumulative probability distribution (cpd).

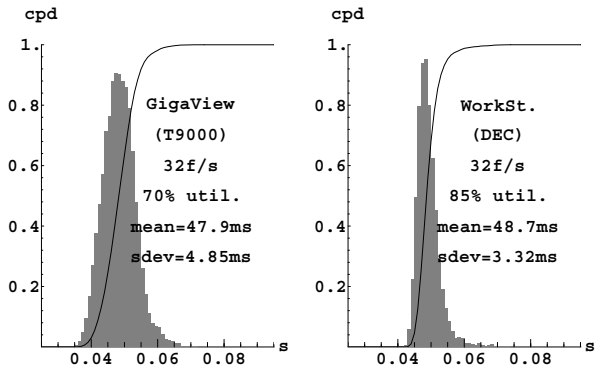

Figure 4 : Single-stream access-time distribution

The two architectures have similar delays : the workstation cluster fast processor makes up for its relatively slow bus. The comparison of the GigaView and the workstation cluster yields a rather unintuitive result : the two architectures have the same delay, but the workstation cluster has the smaller delay jitter. To any user of a workstation with unpredictable response time, this comes as a surprise. The explanation comes from the fact that the workstation cluster bus is a bottleneck. All bus requests are therefore delayed, and hide the jitter due to the disks.

Delay distribution. 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 throughputs up to 30 frames/s, all cpd curves are similar.

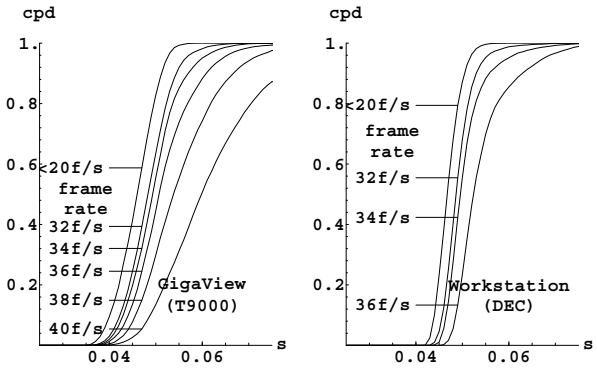

Figure 5 : cpd vs. delay and utilization, for a single stream

The workstation architectures has a small delay jitter but is unable to sustain throughputs above 36 frames/s (14.4MBytes/s). Above 40 frames/s, the GigaView architecture is slowed down by the memory throughput of its server interface processor. Replacing the T9000 by the faster alpha processor would allow the GigaView architecture to sustain throughputs of up to 50frames/s (20MBytes/s).

5.3 Multiple users

In this experiment, the image server supplies three streams. The one-second time-slice request-pattern of stream one (respective two and three) consists of 8 (respective 7 and 6) uniformly distributed frame requests. The one-second time-slice utilization is 46.53% (resp. 53.63%) for the GigaView (resp. workstation cluster). 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.

Figure 6 shows the access delay distribution of the three combined streams, for both architectures. The throughput is 27 frames/s, i. e. 10.8MBytes/s, for a time slice of 776ms. In this experiment, the GigaView (resp. workstation cluster) utilization is 60% (resp. 70%). Stream interactions more than double the maximum access-delay, bringing it to 130ms, compared to the single stream access-delay performance of 50ms.

The multiple-user analysis suggests that streams with different frame rates strongly affect the delay jitter. If absolute delay is of importance, and buffering is not an alternative, it is worthwhile considering whether to constrain frame rates on a parallel image server shared between multiple users to a basic frame rate or an integer fraction of it.

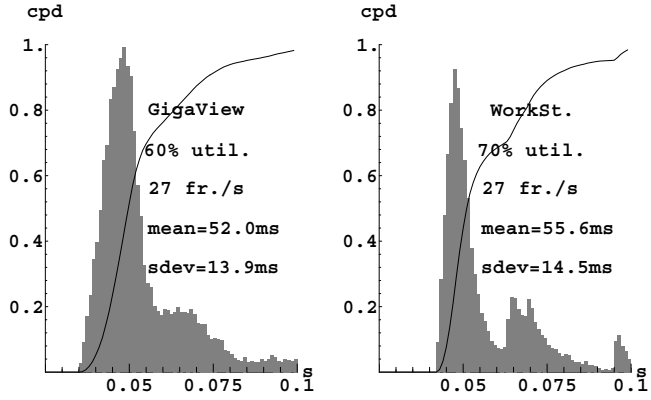

Figure 6 : Delay distribution for multiple streams

6 Conclusion

This contribution compares the image and multimedia performance behavior of a shared-memory-and-bus based multiprocessor multidisk (MPMD) workstation cluster with that of an MPMD architecture having processor-to-processor communication channels (GigaView) instead of global memory or buses. Image window visualization requires reading image extents from disks to the processors' local memory, sending them to the server interface processor and merging them into a single visualization window. Since in the GigaView architecture local disk node processors independently read extents from their disks, no shared resources are required for these operations. The only resource where processing needs to be carried out sequentially is the image part merging process running on the server interface processor. With the workstation cluster architecture however, shared resources are used for nearly every operation: reading from the disks requires copying blocks from the I/O channel to global memory and from there to the processor caches. Image extents need to be transferred through the shared bus to global memory, where they become merged into the desired visualization window. Experimentation and simulations show that the shared bus is the workstation cluster server's bottleneck. In order to achieve a given throughput at the user level, five times that throughput is necessary at the shared bus level. However, the GigaView architecture needs to sustain only a fraction of the user throughput at the level of the disk node processors. At the server interface processor level, a local memory throughput three times as large as the user-level throughput is sufficient in order to receive extents and merge them into a single visualization window. We can therefore conclude that workstation cluster architectures do not perform well for pixmap image access tasks and that the same performance can be obtained at a much lower price with a GigaView architecture based on point to point communication between processors.

Regarding the multimedia performance of both architectures, a dedicated workstation cluster architecture having a 75MBytes/s bus throughput and serving only a single set of requests at the time (no task switches) offers, due to the balancing effect of its shared bus, a lower delay jitter than the GigaView architecture. Nevertheless, the total access delay is slightly longer and its utilization rate higher than the corresponding parameters of the GigaView architecture.

In the case of multiple streams having different, non-commensurable access rates, stream interactions more than double the maximum access delay and are responsible for a delay jitter which is much longer than the mean access delay. Contentions between streams may be reduced by introducing a single basic frame rate for all stream requests and possibly integer subframe rates for lower throughput streams. By appropriately sequencing such multiple frame requests, one would obtain delay jitters close to those of single frame requests.

References

[1] Ann L. Drapeau et al. Raid-II : A high-bandwidth network file server. In *Proc. 21th Int. Symp. Computer Architecture*, pages 234–244, Chicago, Illinois, 1994.

[2] 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.

[3] 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.

[4] 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.