GigaView Parallel Image Server Performance Analysis

Benoit A. Gennart, Bernard Krummenacher, Laurent Landron, Roger D. Hersch
Ecole Polytechnique Fédérale de Lausanne (EPFL)

Abstract. Professionals in various fields such as medical imaging, biology and civil engineering require rapid access to huge amounts of uncompressed pixmap image data. Multi-media interfaces further increase the need for large image databases. 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 and experimentation the behavior of the GigaView under single and multiple requests, and compares it to the behavior of RAID servers. It evaluates image visualization window access times under various parameters such as load factors and the number of cooperating disk nodes. Under single request, the GigaView image server can be modeled as a single high-throughput low-latency secondary storage device. Under multiple requests, the notions of utilization and maximum sustainable throughput define accurately the behavior of the GigaView.

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 increases at a much slower rate. Interactive real-time visualization of full color pixmap image data requires throughputs 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 images stored on disk arrays. Basic services include real-time extraction of image parts for panning purposes, resampling for zooming in and out, browsing through 3-d image cuts and accessing image sequences at the required resolution and speed.

Previous research focussed on increasing transfer rates between CPU and disks by using Redundant Arrays Of Inexpensive Disks (RAID)[2]. Access to disk blocks was parallelized, but block and file management continued to be handled by a single CPU with limited processing power and memory bandwidth.

In a more recent research project [7][5], the RAID concept was further extended to offer very high bandwidth disk arrays directly hooked onto high-speed networks (HIPPI based networks). In this paper, we present a different approach: the multiprocessor multidisk approach we propose aims at associating disks and processors into 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

Proc. The World Transputer Congress, Sept. 1994, in DeGloria et al. editor, 120-135

are highly valuable in the case of image accesses: large pixmap images can be reduced into displayable size images at disk reading speed [4].

In the multiprocessor-multidisk (MPMD) approach [3], 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 (T800 transputer), cache memory (6 MBytes) and one disk (400 to 1000 MBytes).

The authors have implemented an MPMD image server, called the GigaView. It provides through standard host or network interfaces (at the moment SCSI-2, FDDI and ATM in the near future) a throughput of up to 5MBytes/sec., and the ability to browse through images and maps of arbitrary size at the rate of three to four 512-by-512 full color image visualization windows per second.

This contribution analyzes through simulation and experimentation the behavior of image servers (single-disk, RAID, GigaView) under single and multiple requests. Under single request, it demonstrates that, regardless of the architecture and data allocation, the behavior of the GigaView can be modeled as a single high-throughput low-latency disk. Under multiple requests, it shows that the GigaView behaves as a fixed-service-time server.

Section 2 describes two approaches to mass storage servers, the RAID approach, and the MPMD approach. It also introduces MDFS, the multi-dimensional file system used to store images on the multiprocessor- multidisk architecture. Section 3 analyses through simulation and experimentation the single-request performance of the GigaView and compares it to a single disk system running MDFS and a RAID system running MDFS. Section 4 analyzes through simulations the behavior of the GigaView under multiple requests. It compares the GigaView behavior to the behavior of a reference fixed-service-time server.

The mass storage server studied in this publication stores images of arbitrary sizes, i.e. two- or three-dimensional arrays of pixels, each pixel representing its color as a given number of bytes (assumed to be 3 in our experiments). Each image stored on the server is divided into extents, or rectangular image portions. Limited size visualization windows (generally 512-by-512 pixels) are requested by client applications.

2. Image storage architectures

This section describes the RAID approach, the Lancaster Multimedia Network Interface (MNI) approach, as well as the software and hardware concepts of the GigaView architecture (sections 2.3. and 2.4.).

2.1. Raid storage server

Disk arrays are often referred to as RAID (Redundant Arrays of Inexpensive Disks). The original RAID research considered [6][9] five types of disk array organizations. The ideal RAID level for imaging should provide high data transfer rates for large image files, whose extents are striped onto the disk array. It should also provide large storage capacities while extracting a minimal premium for redundancy. RAID level-5 disk arrays fulfill these goals. For the RAID experiments conducted in this study, the authors considered the RAIDER-5 (4+1)-disk system located at the Rural Engineering Department of EPFL. The RAIDER-5 system is a RAID level-5 disk array. The major difference between the RAID approach and the approach taken in this paper is the ability of the GigaView to process data locally on the server.

2.2. Lancaster storage server

An experimental storage server has been built [1][8] at Lancaster to investigate the techniques required to support continuous media. The Lancaster Continuous Multimedia Storage Server (CMSS) consists of a standard Multimedia Network Interface (MNI). The MNI is implemented using transputer technology and is connected on one hand to a high-speed network, and on the other hand to two stripe disks and one directory disk. There are three differences between the Lancaster system and the Gigaview : (a) the GigaView architecture offers more local processing power and more throughput ; (b) the GigaView supports random access to images efficiently ; (c) the GigaView multi-dimensional file system (see section 2.3.) guarantees excellent access time to image visualization windows, regardless of image size.

2.3. Multi-Dimensional File System (MDFS)

Image access characteristics are known : client workstations generally require rectangular portions of pixmap image files, referred to as visualization windows. Storing a large image scanline by scanline leads to poor access times. One must either fetch large horizontal stripes of images, leading to unacceptably large transfers of data, or fetch the visualization window scanline by scanline, leading to a large number of disk-seek operations, each taking around 10msec. For instance, accessing a 600-by-400 window in a 5000-by-5000 image scanline by scanline takes between 4 and 6 sec.

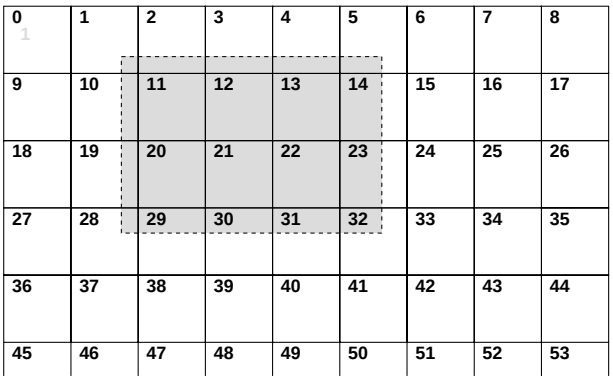

FIGURE 1. Division of an image into extents

A better approach to accessing visualization windows in a large image is to partition the image into rectangular extents (Figure 1). In such a format, accessing a visualization window consists of fetching only the extents covered by the window. This minimizes both the number of seek operations and the data transferred. Accessing the same 600-by-400 window in an image stored as 128-by-128 extents on a standard disk requires 20 extent accesses, and can be performed in a little over a second. The Multi-Dimensional File System (MDFS) stores 1-dimensional (1-D) files, 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 [4]. For each file, MDFS stores metadata such as image file segmentation parameters (extent size, number of striping disks, pixel size), index information such as the disk identifiers on which the file is striped, the position on each disk of the corresponding file local extent table and, in each file local extent table, pointers to the local file extents.

Experimental results confirm the benefits of storing a large image using MDFS. Figure 2 shows the time required to transfer visualization windows of varying sizes from a Sparc-

Classic disk to memory, for five kinds of extents : 200-by-ImageWidth extents, 1-image-scanline-per-extent extents, 50-by-50 extents, 100-by-100 extents, and 150x150 extents. The SparcClassic local disk is a 1GB Quantum with SCSI interface, 10msec. seek-time and 2.93MBytes/sec. sustained throughput. The size of the image from which the visualization windows are requested is 3072-by-2048 3-byte pixels.

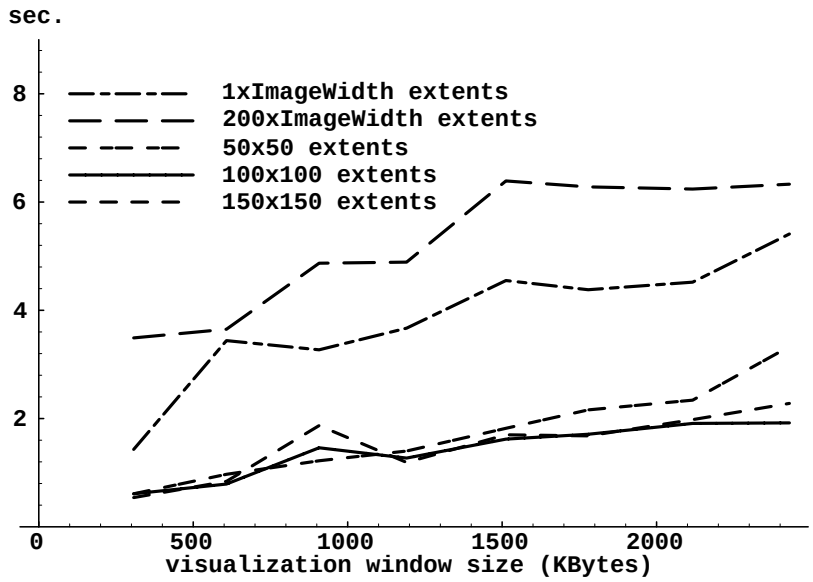

FIGURE 2. Running MDFS on a single disk

Once the data is cached, MDFS is even more superior to other file organizations (Figure 3). Indeed, MDFS accesses very little data outside the visualization window, and therefore makes the best use of the cache.

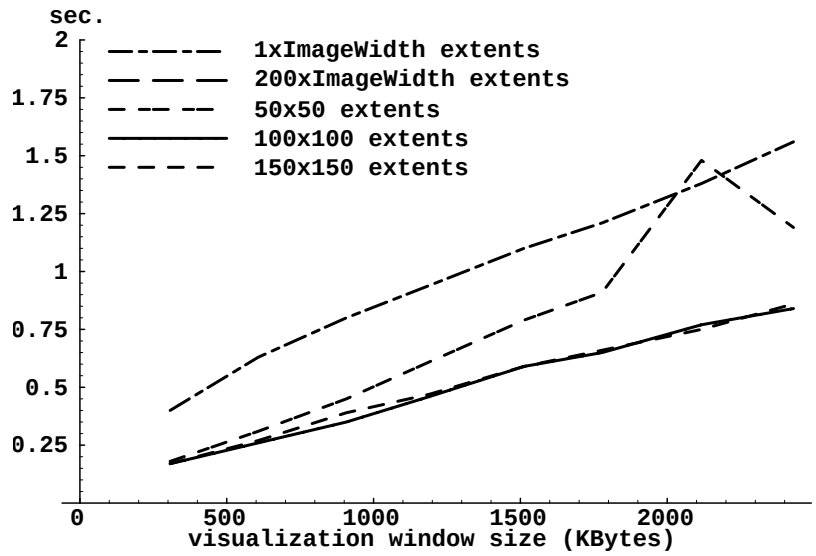

FIGURE 3. Running MDFS on a single disk : cached data

The best overall performance is obtained by the 100-by-100 extent allocation ; too small an extent increases the overhead due to accessing extents, and too large an extent increases the amount of data to be fetched to display a visualization window of a given size. The 200-by-ImageWidth extents yield such poor performance (between 3 and 5 sec. access-time) that their data is not shown in Figure 3.

2.4. GigaView Multi-Processor Multi-Disk Architecture

The parallel image server consists of a server interface processor providing the network interface, disk node processors used for disk access, image part extraction and image reduction, as well as an array of disk nodes, each disk node being connected to one disk node processor (Figure 4). 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 number of disks in the architecture can be varied, by connecting more than one disk node to each server interface processor link.

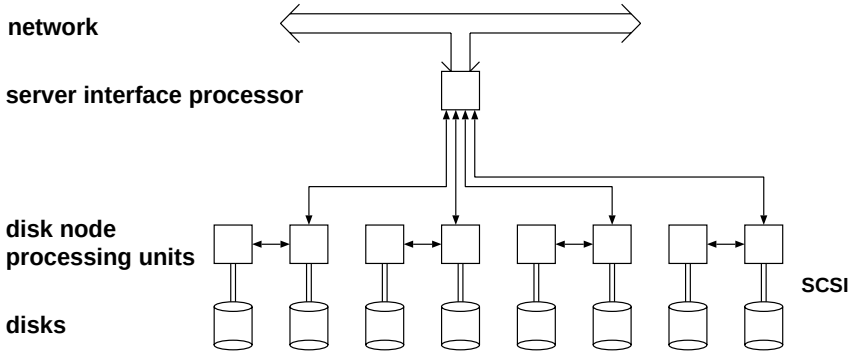

FIGURE 4. GigaView 8-disk architecture

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.

D0	D1	D2	D3	D4	D5	D6	D7	D0	D1	D2
D3	D4	D5	D6	D7	D0	D1	D2	D3	D4	D5
D6	D7	D0	D1	D2	D3	D4	D5	D6	D7	D0
D1	D2	D3	D4	D5	D6	D7	D0	D1	D2	D3
D2	D3	D4	D5	D6	D7	D0	D1	D2	D3	D4

FIGURE 5. Extent allocation with a row offset of 3

Image access performances are heavily influenced by how pixmap images are distributed onto a disk array. The k disk nodes in the disk array selected for storing an image file are numbered from 0 to k-1. Image file extents are mapped sequentially on one disk after the other. At image storage time, image size and number of disks are known. The image partitioning problem is reduced to the problem of finding the proper extent row offset, that is, the

difference in disk index between two extents of the same image extent column. The extent row offset should ensure that extents covered by the same visualization window are distributed as uniformly as possible on the set of available disk nodes [4].

Figure 5 shows that, for an 8-disk architecture, a row offset of 3 provides an excellent distribution of disk accesses : no disk is accessed more than 3 times. Simulations confirm that an extent row offset of 3 is effective for an 8-disk architecture regardless of the visualization window size and position. Previous results [3] have shown that the extent size should be no larger than a quarter of the average visualization window size, no smaller than 25 KBytes and that the extent row offset and the number of disks in the architecture should be mutually prime.

3. Performance modeling under single-request

This section describes how to model the performance of parallel storage servers under single requests. The analysis is based on simulation results. The validity of the simulation results is confirmed by experimental data. Section 3.1. describes the simulation model ; section 3.2. analyzes the single-request performance of the GigaView architecture ; section 3.3. shows the effect of the number of disks on the architecture performance ; section 3.4. compares the performance of 3 actual parallel-server architectures (Raid-III, Raid-V, GigaView) under single request.

3.1. Simulation model

The following program in free-form programming language describes the modeled behavior of the GigaView.

```
component GigaView is
     InterfaceT Interface ;
     LinkT DownLink[NUMBER OF LINKS] ;  -- from interface to disks
     LinkT UpLink[NUMBER OF LINKS] ;    -- from disks to interface
     DiskT Disk[NUMBER OF DISKS] ;
     procedure Read (VisualizationWindowT window) ;
end GigaView ;

procedure GigaView.Read (VisualizationWindowT window) is
begin
     Interface.Decompose (window, ExtentRequests) ;
     foreach ER in ExtentRequests do
          -- ER : extent request, i.e. image, size, position
          DownLink[ER.Link].Transfer (ER) ;
          Disk[ER.Disk].Access (ER, Extent) ;
          -- Extent : i.e. image, size, position, pixmap data
          UpLink[ER.Link].Transfer (Extent) ;
          Interface.Merge (Extent, window) ;
     end foreach ;
end GigaView.Read ;
```

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 where the extent is located. The extent is fetched from the disk and transferred up a transputer link back to the

interface processor, where it is merged with the other extents to form the visualization window.

The simulation model assumes that the disk access-time, the transputer-link transfer-time, and the transputer memory-to-memory copy-operations obey simple linear formulas of the form Delay = Latency + (DataSize / Throughput) ; it also assumes that the time required to decompose a visualization window into extent requests is linear in the number of extent requests. To validate the simulation model, we benchmarked all components of a 4-disk architecture, as well as the whole system, for single and multiple requests. Whenever the experimental data is available, we plot it alongside the simulation data (figures 6 and 13). For both single- and multiple-request analysis, the experimental data confirms the validity of our model.

3.2. Latency and throughput

This section shows by simulation that it is possible to describe the behavior of a parallel storage server using two numbers, latency and throughput. This is similar to the way secondary storage devices are described by two numbers, 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 (Mathematica), and get a formula of the type :

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

The GigaView architecture performance is sensitive to the extent allocation scheme. In particular, the extent size and the row offset have to be chosen carefully to reach the best performance. As shown in section 2.4., an extent size of 128-by-128 pixels and an extent row offset of 3 are effective for a wide range of visualization window sizes and optimum for a visualization window size of 512-by-512 pixels.

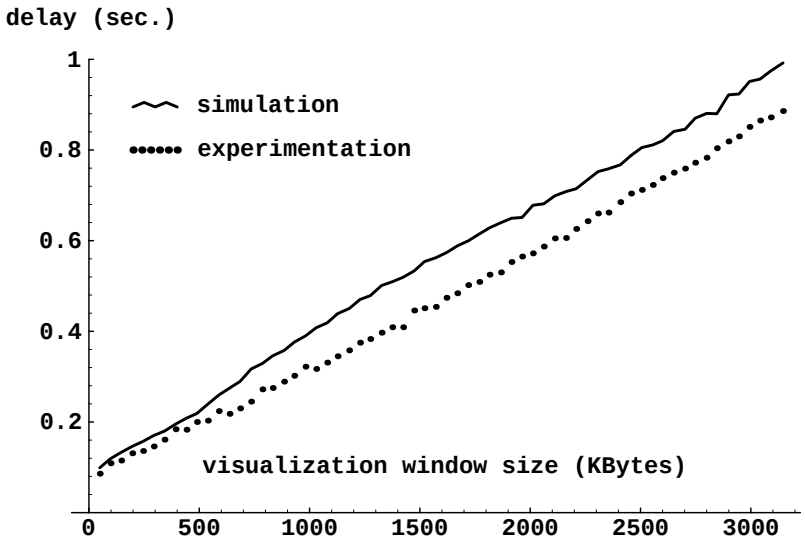

FIGURE 6. Latency and throughput of a 4-disk-node architecture

The performance of the GigaView is also sensitive to the position of the visualization window. However, the analysis cannot make any assumptions about the position of the visualization window. To take into account this effect, the visualization window is accessed 40

times for each size, at random positions, and the resulting delays are averaged. The average gives the access time for a given visualization window size. Simulations show that increasing the number of window positions does not significantly improve the precision of the delay measurement.

The linearization approach has proved particularly effective, regardless of the data allocation and the architecture of the system. Figure 6 shows both simulation and experimental results (ragged and dotted lines). The simulated architecture consists of 4 disks and 4 links between server interface processor and disk nodes. The T800 transputers have a memory bandwidth of 18MBytes/s and each communication link has a throughput of 1.6MBytes/sec. The disks are Quantum-SCSI2, whose seek-time and throughput have been measured experimentally at being respectively 20msec. and 2.28MBytes/sec.

The simulated delay can easily be linearized. The linearization shows that over a wide range of window sizes, the 4-disk-node GigaView architecture can be modeled as a single high-performance disk system with a latency of 64msec., and a throughput of 3.88MBytes/sec. The black dots in Figure 6 represent experimental points corresponding to the actual measured performance of the GigaView under the same circumstances. The measured performance of the GigaView is actually better than the simulated performance of the GigaView. This can be traced to the fact that extents are allocated contiguously on disk. Therefore, on the running prototype system, the seek-time required to locate the first extent is longer than the seek-time required to locate subsequent extents.

3.3. Effect of the number of disk-nodes

Using the linear model of the performance of the GigaView, it is easy to demonstrate the effect of the number of disk-nodes in the architecture on the performance of the system. Figure 7 shows the simulated access-time to a visualization window of increasing sizes for 4 architectures : 1-disk-node, 2-disk-node, 4-disk-node and 8-disk-node architecture.

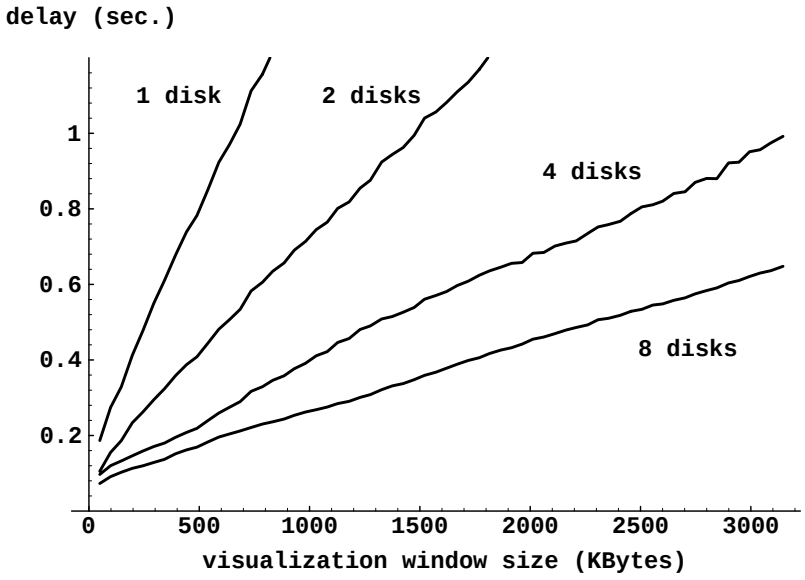

FIGURE 7. Latency and throughput vs. number of disk-nodes (T800-based architecture)

Figure 7 shows that latency decreases and throughput increases as the number of disk-nodes increases. Beyond 8 disk-nodes, adding more disk-nodes ceases being beneficial, since link communication bandwidth limits overall performance.

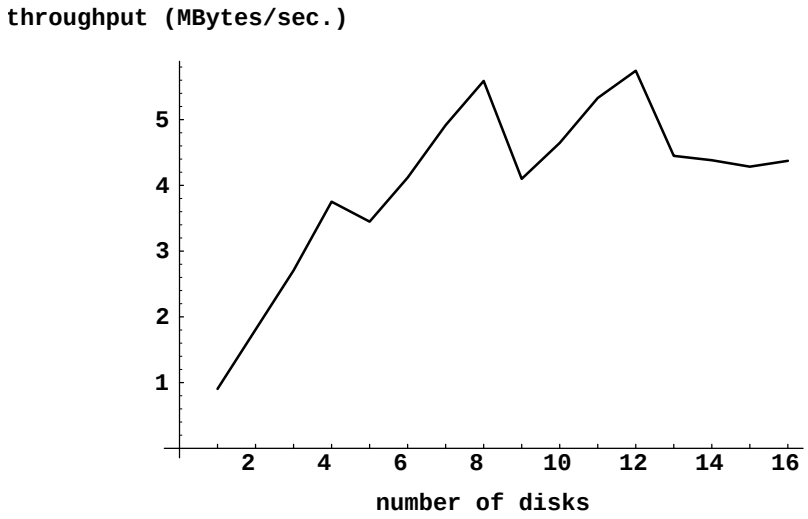

FIGURE 8. Throughput vs. number of disk-nodes (T800-based architecture)

Figures 8 and 9 respectively plot the linearized throughput and the latency versus the number of disk-nodes for this T800-based transputer architecture. The peak throughput is reached for an 8-disk-node architecture, and the latency remains around 80msec for all architectures with more than 4 disk-nodes. The throughput curve reaches local maxima when the number of disk-nodes is a multiple of the number of links (4, 8, 12). The single-request study seems to suggest that a GigaView architecture with more than 8 disk-nodes is not cost-effective. The multiple request study will show a different result.

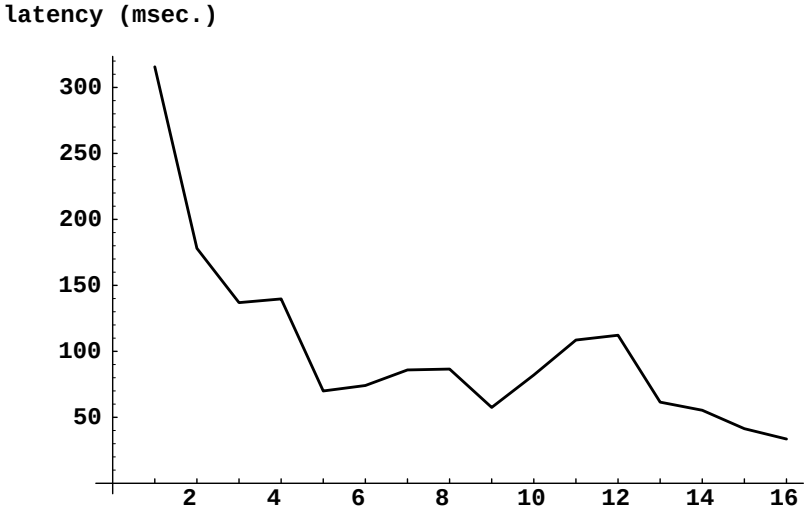

FIGURE 9. Latency vs. number of disk-nodes (T800-based architecture)

It is possible to get a precise idea of the maximum number of disk-nodes the architecture effectively supports by carrying out a single single-request experiment. The key concept is that of component utilization, defined as the ratio between a given component's active-time and the total simulation time.

The simulation consists of requesting a single 512-by-512 3-byte-pixel visualization window, on a 4-disk-node T800-based architecture. Table 1 shows the utilization of each component of the 4-disk-node architecture, as well as the average utilization of each class of components (interface processor, links, disk-nodes). The utilization data are part of the sim-

ulation results. In a 4-disk-node architecture, the average disk-node utilization is 86, the links are 42%-utilized, and the interface processor is 33%-utilized. In the 4-disk architecture, the ratio between disk-node- and link-utilization is 2. This suggests that an 8-disk-node architecture provides an equal utilization of disk-nodes and links (Table 2). Above 8 disk-nodes, the transputer links are more utilized than the disks. The links become the limiting factor in the architecture, and therefore adding more disks to the architecture does not provide any performance improvement.

	0	1	2	3	average
disk-node	0.95	0.83	0.83	0.83	0.86
link	0.42	0.42	0.42	0.42	0.42
interface processor	0.33				0.33

Table 1: Utilisation of GigaView components (T800 processors, 4 disk-nodes, 512x512 visualization window size)

Figure 8 and table 2 confirm that the best performance (in terms of throughput) is achieved by the balanced architecture, i. e. the architecture for which the links and disks are equally utilized. Table 2 also shows that the maximum component utilization decreases significantly when stepping up the architecture from 4 to 8 disk-nodes. This explains why the delay of an 8-disk architecture (0.218s for a 512-by-512 3-byte-pixel visualization window) is more than half the delay of a 4-disk architecture (0.332s). For small visualization windows, changing the data allocation scheme to improve the utilization by decreasing the extent size does not improve performance : the overhead due to the larger number of extents negates the effect of the improved data allocation.

	0	1	2	3	4	5	6	7	average
disk-node	0.62	0.63	0.63	0.63	0.84	0.64	0.63	0.64	0.66
link	0.55	0.63	0.76	0.68					0.66
interface processor	0.51								0.51

Table 2: Utilisation of GigaView components (T800 processors, 8 disk-nodes, 512x512 visualization window size)

3.4. Experimental results

This section compares the access delays of four storage systems. The first configuration is an actual SparcClassic workstation and its local disk and the second configuration is an actual RAIDER-5 system connected to a Sparc server 1000. The third system is an actual RAID level-3 system connected to a Cray Y-MP. The fourth system is an actual 4-disk-node GigaView system. The SparcClassic local disk is a 1GB Quantum with SCSI interface, 10msec seek-time and 2.93MBytes/sec. sustained throughput. The RAIDER-5 system is a RAID-5 architecture consisting of (4+1) WREN-9 disks having a latency of 12.9 msec. and a wide SCSI-2 interface. The RAID level-3 system consists of 10 disks (8 + 2 spare) Hitachi DK-516-15.

The experiment consists of transferring visualization windows of increasing sizes from disk(s) to host memory, and measuring the transfer times. All architectures run MDFS, the multi-dimensional file system. The image from which the visualization windows are selected is 3072-by-2048 3-byte pixels in size, and is divided in 128-by-128 extents. On the first three configurations (single-disk station and the two RAID servers), the entire data is experimental.

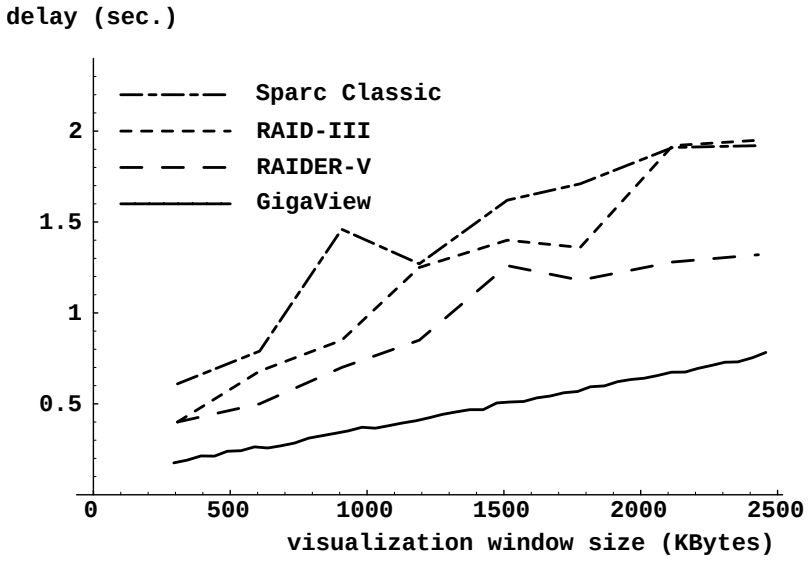

FIGURE 10. Experimental results

For the GigaView performance measurements, it is assumed that transferring a visualization window from disk to host is a two-stage pipeline. The first stage of the pipeline transfers extents from the disks to the GigaView server interface processor memory. The second stage transfers image segments consisting of rows of extents from the server interface processor memory to the host memory. The access delay is a combination of (1) actual delays measured on the GigaView system (transfer of the whole visualization window between the disks and the GigaView interface processor) and (2) a conservative estimate of the transfer delay of one horizontal row of extents between the GigaView interface processor and the host memory. The formula giving the estimate of the SCSI bus transfer time is :

$$\frac{RowOfExtentSize}{SCSIThroughput} = \frac{\left\lceil \frac{ImageWidth}{ExtentWidth} \right\rceil \cdot ExtentSize}{5 \cdot 10^6}$$

where the curtailed bracket symbol represents the round-up operation. The fact that the GigaView performance is superior to the RAID systems performance can be traced to the fact that the GigaView has an excellent control over extent allocation, which could not be achieved on the tested RAID-III and RAIDER-V systems. To achieve the best visualization window access times, it is necessary to control precisely the disk allocation of each image extent. Another important point is that the high-throughput low-latency behavior of the GigaView is achieved despite the comparatively longer seek-time of the Quantum-SCSI2.

4. Multiple-request behavior

This section describes the behavior of the GigaView under multiple requests. In order to provide a reference point, this study compares the behavior of the GigaView under multiple-

request to the behavior of an abstract fixed-service-time server. It shows that, due to internal pipelining, the GigaView sustains higher throughput than the fixed-service-time server. The amount of additional throughput depends on the single-request utilization of the disk array. Section 4.1. describes the conditions under which the simulations are conducted. Section 4.2. demonstrates the validity of our simulation model under multiple requests. Section 4.3. simulates the GigaView under multiple requests, and compares its behavior to the reference server. It also evaluates the effect of the number of disks on the performance of the architecture.

4.1. Simulation characteristics

Requests to the GigaView represent a Poisson process. This means that individual requests are independent and that the number of requests in a given time interval only depends on the length of that interval. The interval between requests therefore follows an exponential distribution. The load on the system is expressed in terms of requested throughput. In our simulations, all users request a 512-by-512 3-byte-pixels visualization window (786 KBytes). Therefore, a requested throughput of 3MBytes/sec. corresponds to 4 window requests per second. The Poisson process hypothesis also ensures that, for a given load, the number of users requesting windows from the system has no effect. Only the requested throughput affects the average response time of the system.

For a given system architecture, each simulation 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 simulation is the delay average over the 5000 requests.

4.2. Experimental model validation

To validate our simulation model under multiple requests, we measure the access-times of an actual GigaView architecture consisting of 4 Quantum-SCSI-2 (23msec average seek-time and 2.28MBytes/sec. throughput) and 4 T800 processors (memory bandwidth 18MBytes/s ; link bandwidth 1.6MBytes/sec), and compare it to the simulation of the same architecture. In this experimental setup, the requests for visualization windows represent a Poisson process with the same statistical properties as a sequence of requests generated during a multiple-request simulation.

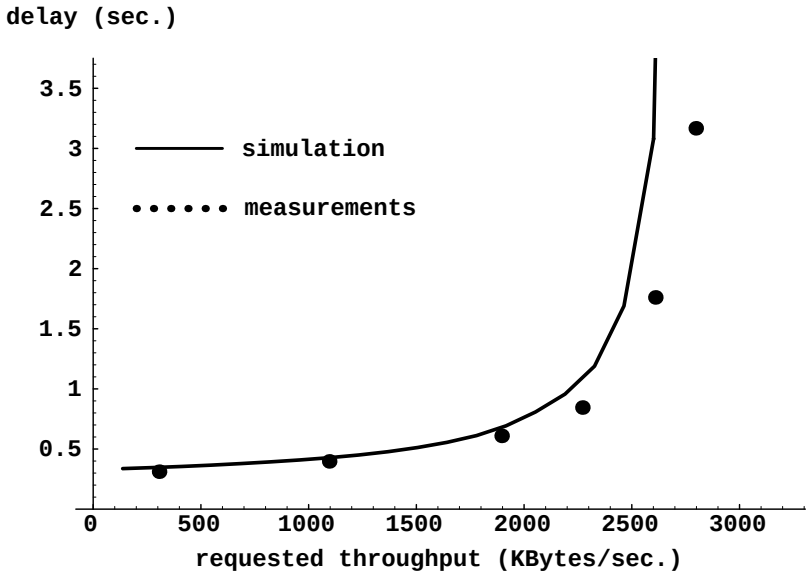

FIGURE 11. Comparing simulation and experimental results (T800-based GigaView architecture)

Figure 11 shows the average access delays of both the simulated architecture (continuous line) and the actual architecture (dots). It confirms the validity of our simulation model, both qualitatively and quantitatively. The multiple-request performance of the actual system is superior to the performance of the simulated architecture, for the same reason that the single-request performance was superior to the single-request performance of the simulated architecture : the seek-time required to locate the first extent in an image is longer than the seek-time required to locate subsequent extents (see section 3.2.).

4.3. T9000-based GigaView architecture

In this section, we study the behavior of the GigaView under multiple requests. The architecture consists of T9000 transputers and Quantum-SCSI-2 disks. Since the T9000 transputers were not yet available at the time of submission, their performance was conservatively estimated at 36MBytes/s memory bandwidth and 8MBytes/s link transfer rate. The Quantum-SCSI-2 latency and throughput are measured experimentally at 20msec. and 2.23MBytes/sec. The fixed-service-time server provides a reference point for the GigaView simulations. Its only property is its service-time, equal to the service-time of a single visualization window request. Requests to the reference server follow the same distribution as requests to the GigaView. For example, a T9000-based 4-disk-node GigaView architecture satisfies a 512-by-512 3-byte-pixel visualization window request in 0.305 sec. The maximum throughput sustainable by the fixed-service-time server is (MST = maximum sustainable throughput) :

$$MST = \frac{SingleRequestSize}{SingleRequestDelay} = \frac{768KBytes}{0.305s} = 2.62MBytes/s$$

Figure 12 shows the performance results of the GigaView. The continuous line represents the GigaView performance (delay average), whereas the crosses represent the performance of the fixed-service-time server (delay average).

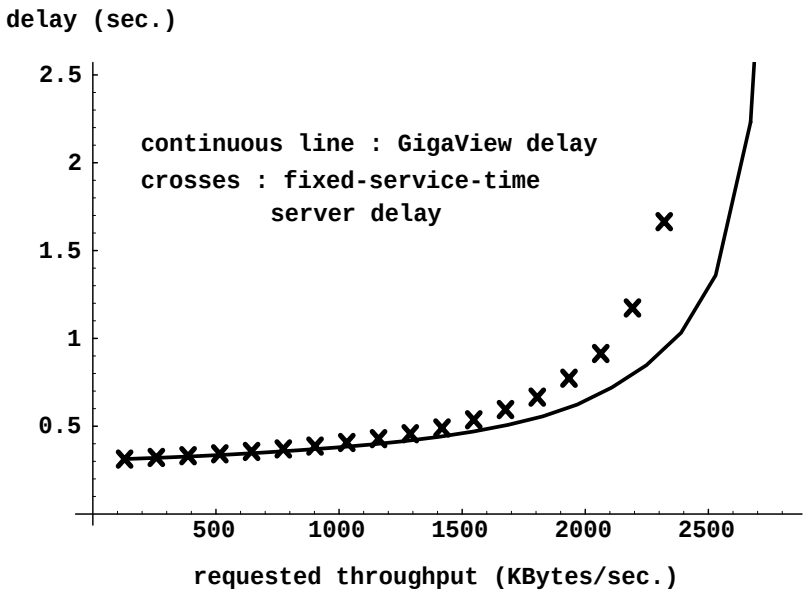

FIGURE 12. GigaView performance under multiple requests (T9000-based architecture)

Figure 12 shows that the performance of the GigaView is superior to the performance of a fixed-service-time server. This result is not difficult to explain. During a single-request

experiment, no component of a 4-disk-node GigaView architecture is used more than 90 of the time. Therefore, under multiple-request, some amount of internal pipelining occurs, making the GigaView able to sustain higher loads than the fixed-service-time server.

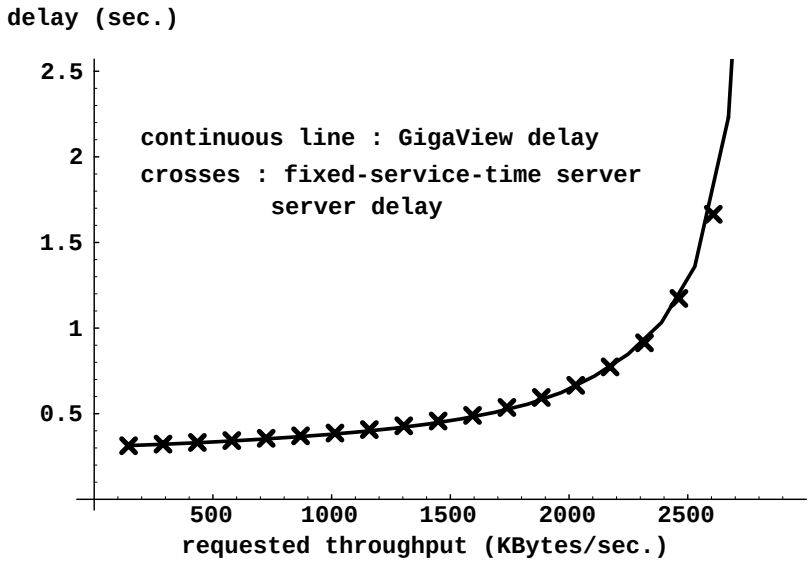

FIGURE 13. Matching reference server and GigaView (T9000-based architecture)

One can match the behavior of the fixed-service-time server and the GigaView by scaling the x-axis of the fixed-service-time server performance curve by a factor equal to the inverse of the single-request utilization of the GigaView (Figure 13). This suggests that the GigaView maximum sustainable throughput (MST) must be defined as :

$$MST \cong \frac{SingleRequestSize}{SingleRequestDelay} \cdot \frac{1}{SingleRequestUtilization}$$

In this formula, the single-request size is a simulation parameter ; the single-request delay and utilization are simulation results. The formula holds true regardless of the single request size. A single single-request simulation allows to evaluate the maximum sustainable throughput of a given architecture.

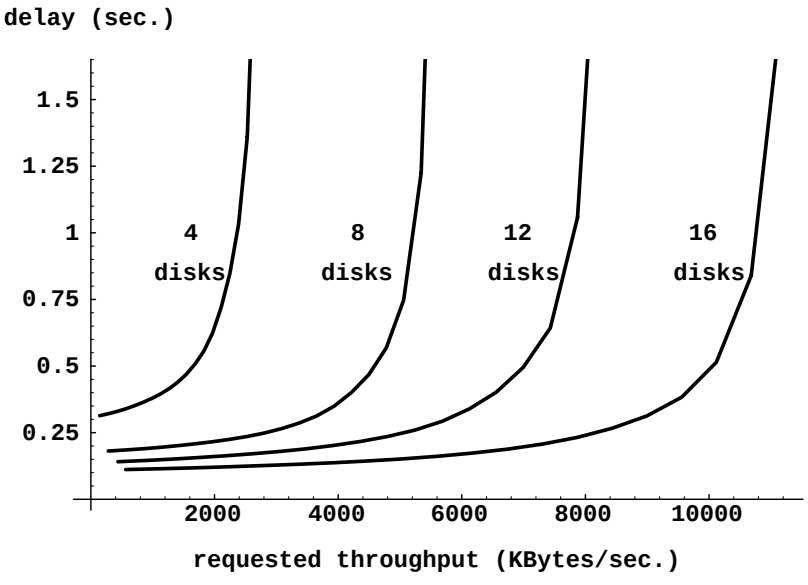

FIGURE 14. Effect of the number of disk-nodes (T9000-based architecture)

Effect of the number of disk-nodes. Figure 14 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.

Consider a requested throughput of 6 MBytes/sec. The average delay for a 12-disk-node architecture is around 400msec, whereas a 16-disk-node architecture satisfies requests on average within 200msec, i.e. an improvement by a factor of 2. This seems to be in contradiction with the single request analysis of the same architecture.

The single request analysis applied to a T9000-based architecture (Table 3) shows that the maximum throughput is reached for a 12-disk-node architecture. The 16-disk-node architecture does offers very little benefit over the 12-disk-node architecture, in terms of single-request throughput or access delay. The major difference between the two architectures lies in the utilization of 12-disk and 16-disk architectures under single-request. In a 12-disk-node architecture, disk-node components are utilized on average 76% of their time, and in a 16-disk-node architecture, they are used on average 61% of their time.

number of disks	4	8	12	16
delay (983KBytes,ms)	354	89	143	135
latency (ms)	123	75	95	44
throughput (MB/s)	4.01	7.97	12.8	11.8
utilization (%)	93	87	76	61

Table 3: GigaView single-request analysis (T9000-based architecture)

Using the maximum sustainable throughput formula introduced earlier, we find that the maximum sustainable throughputs are 2.98MBytes/s (respectively 5.97MBytes/s, 9.04MBytes/s, 11.94MBytes/s) for a 4-disk (respectively 8-disk, 12-disk, 16-disk) architecture. Although the single request throughput does not increase above 12 disks, the maximum sustainable throughput under multiple requests increases linearly with the number of disks, for up to 16 disks. Above 16 disks, the interface processor becomes saturated, and the maximum sustainable throughput does not increase anymore.

5. Conclusion and future work

This paper shows through simulation and experimentation that the GigaView multi-processor multi-disk mass storage server is able to sustain throughputs of up to 5MBytes/s for a 4-disk-node T800-based architecture, and of up to 12MBytes/s for a 16-disk-mode T9000-based architecture, while accessing random visualization windows. The access delay is insufficient to describe the architecture performance. Under single request, the concepts of latency and throughput provide a much better picture of the architecture performance. Simulations and experimentation show that it is accurate to model the GigaView as a single high-throughput and low-latency server. Under multiple request, the concept of utilization is essential in estimating the maximum sustainable throughput. The multiple-request study suggests that, although adding disk may not improve the single-request throughput, it improves the maximum sustainable throughput under multiple requests.

Future work will focus on designing a flexible processing scheme for downloading custom code to the disk-node servers, thereby combining high-throughput data access from the disks and high-performance preprocessing operations. By adding features to synchronize multiple heterogeneous streams of data (e. sound and image files), and designing new data

allocation algorithms taking into account the specificity of each stream, the parallel image server presented in this paper will become a parallel multimedia server.

References

[1] 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.

[2] A. Chen and D. A. Patterson. Maximizing performances in a striped disk array. In *Proceedings IEEE International Symposium on Computer Architecture*, pages 322-331, Seattle, 1990.

[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] 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.

[5] 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*, 1993.

[6] R. H. Katz, G. A. Gibson, and D. A. Patterson. Disk system architecture for high performance computing. *Proceedings of the IEEE*, 77(12):1842-1858, December 1989.

[7] 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.

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

[9] D. A. Patterson, G. A. Gibson, and R. H. Katz. The case for RAID : redundant arrays of inexpensive disks. In *Proceedings ACM SIGMOD Conference*, pages 106-113, Chicago, IL, May 1988.