Parallel Storage and Retrieval of Pixmap Images

Roger D. Hersch
Ecole Polytechnique Fédérale de Lausanne
Lausanne, Switzerland

Abstract

Professionals in various fields such as medical imaging, biology and civil engineering require rapid access to huge amounts of uncompressed pixmap image data. To fulfill these requirements, a parallel image server architecture is proposed, based on arrays of intelligent disk nodes, with each disk node composed of one processor and one disk. This contribution shows how images can be partitioned into extents and efficiently distributed among available intelligent disk nodes. The image server's performance is analyzed according to various parameters such as the number of cooperating disk nodes, the sizes of image file extents, the available communication throughput and the processing power of disk node and image server processors. Important image access speed improvements are obtained by image extent caching and image part extraction in disk nodes. With T800 transputer-based technology, a system composed of eight disk nodes offers access to three full-color 512x512 pixmap image parts per second (2.4 megabytes per second). For the same configuration but with the recently announced T9000 transputer, image access throughput is eight images per second (6.8 megabytes per second).

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 requires a throughput of two to ten megabytes per second. Parallel input/output devices are required to access and manipulate, at high speed, image data distributed on disk arrays.

A high-performance, high-capacity image server should 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, re-sampling for zooming in and out, browsing through 3-d image cuts and accessing image sequences at the required resolution and speed.

Previous research was focused on increasing transfer rates between CPU and disks by using *Redundant Arrays Of Inexpensive Disks (RAID)* [1, 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.

To access large quantities of pixmap image data at a throughput of two to ten megabytes per second, a multi-processor-multidisk (MPMD) approach is proposed. Pixmap image data is partitioned into rectangular extents, each extent having a size which minimizes global access time. 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 (six megabytes) and one disk (400 to 1000 megabytes). The hardware part of this architecture is similar to the one used in the DataMesh project [4].

The proposed parallel image server architecture includes an array of disk nodes offering parallel image storage, image-handling processors dedicated for the storage and processing of image parts and a network interface. This contribution discusses how images can be partitioned into extents and efficiently distributed among disk nodes. It analyzes the performance of the system according to various parameters such as the number of cooperating disk nodes, the size of the image file extent, the effect of file image caching, and the type of available processing and communicating capabilities. Performance figures quoted hereafter refer to storing and accessing uncompressed images.

This contribution does not discuss the problem of disk crash recovery. However, RAID-5 redundancy schemes [2] could be incorporated onto the proposed MPMD architecture.

Image Server Architecture

The image server comprises a network interface, a single or several image-handling processors used for image assembly and processing as well as an array of disk nodes (Figure 1). One image-handling 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

1051-9173/93 $3.00 ©1993 IEEE

12th IEEE Symposium on Mass Storage Systems, Montherey CA, April 1993, Digest of papers, IEEE Computer Science Society Press, 221-226

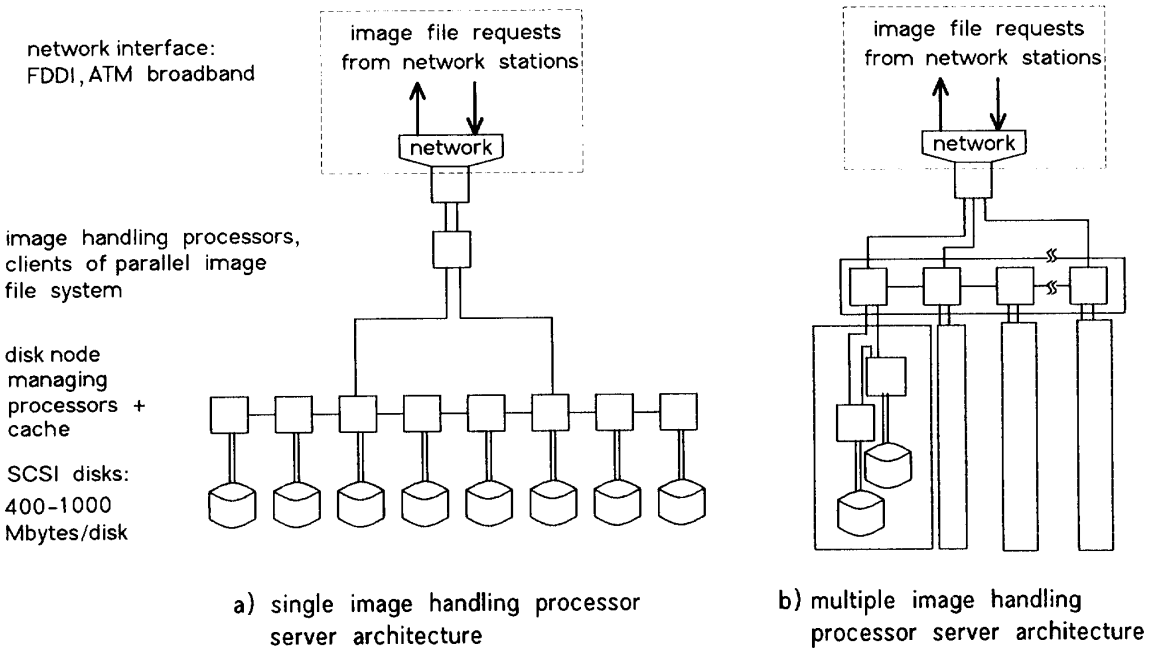

Figure 1. Image server architecture.

serving processes are responsible for serving extent access requests, for maintaining the free block lists and for managing local extent caches. Image processing tasks required for image presentation such as re-sampling, filtering and adaptation of gray levels may be located on interface processors and on disk node processing units.

Extent Size Computation

Image access performance is heavily influenced by the way in which pixmap images are distributed onto a disk array. Image access characteristics are known: client workstations generally require rectangular portions of pixmap image files. Therefore, image file data is partitioned into rectangular extents. To simplify the file system managing parallel storage of files on multiple disk nodes, extents are numbered sequentially from left to right and from top to bottom. 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 one to one in modulo-k mode to disk nodes.

At image storage time, image size, visualization window size and number of disks are known. The image partitioning problem is reduced to the problem of finding a horizontal extent size which ensures that extents lying within the visualization window are distributed as uniformly as possible on the set of available disk nodes (Figure 2). The chosen extent size should be relatively close to an initially given suitable extent size (which, by simulation, is known to provide high parallel disk throughput).

For an initially given extent size, the size of the visualization window determines the number of extents contributing to the window content. If the number of disks is equal to or higher than the number of window extents, the offset between disk numbers from one extent row to the one beneath should be equal to the number of horizontal extents covering the window width. If the number of disks is smaller than the number of window extents, a downscaled visualization window is considered which contains as many extents as there are available disks. The horizontal offset between disk numbers from one extent row to the next should be close to the number of horizontal extents covering the reduced window width. If more than nine extents contribute to a given visualization window, uniform distribution of extents on contributing disks can be ensured if the chosen offset and the number of disks are mutually prime if, for example, their greatest common divisor is no higher than one.

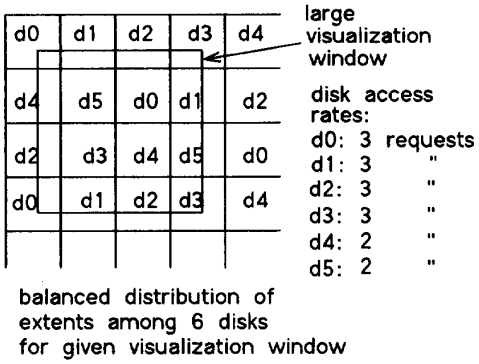

Figure 2. Example of extent distribution among disk nodes.

The desired offset is obtained by varying the extent width. If the desired offset is larger than the one given by the initial extent width, horizontal extent size reduction will increase the positive disk number offset between one extent line and the next. If the desired offset is smaller than the actual one, a horizontal extent size increment will reduce it (Figure 3). Computation of extent size for k disks must make use of modulo-k arithmetic: a positive offset can be increased until k-1, further extent size reduction resets the offset to zero. Furthermore, required offsets are given in absolute values, they can be either positive or negative. One should choose as a desired offset the one that will produce the smaller extent size difference (size enlargement or size reduction).

Initial distribution:

nb of disks: 6
image width: 2048 pixels
extent size: 256 x 256
resulting offset: 2

e0 d0	e1 d1	e2 d2	e3 d3	e4 d4	e5 d5	e6 d0	e7 d1
e8 d2	e9 d3	e10 d4	e11 d5	e12 d0	e13 d1	e14 d2	e15 d3

Distribution for offset = 1
new extent width: 310

e0 d0	e1 d1	e2 d2	e3 d3	e4 d4	e5 d5	e6 d0
e7 d1	e8 d2	e9 d3	e10 d4	e11 d5	e12 d0	e13 d1

Figure 3. Horizontal extent size enlargement reduces offset.

The newly obtained extent width generates the required offset between disk numbers in successive extent rows. However, this width may differ considerably from the initially given extent width. As a consequence, the given visualization window may become covered by more or less extents. Therefore, an iterative process follows, which repeats the complete algorithm several times. This process generally converges after one or two iterations.

Image Access Characteristics

Most slide and page scanners work at a resolution of approximately 3000x4000 pixels. Such a resolution provides both a general view of the image obtained by subsampling or a detailed view of image parts. Windowing systems offer visualization windows of variable sizes: a typical window size may be 512x512 pixels. Such a window may be enlarged to 1024x1024 or reduced to 256x256 pixels.

The considered architecture assumes that the full image visualization window is divided among one or several image-handling processors. Each image-handling processor is linked by two communication channels to a variable number of interconnected disk nodes. For the sake of simplicity, we will analyze architectures having either a single (Figure 1a) or a set of four image-handling processors (Figure 1b). The following questions arise:

- What are optimal extent sizes for given visualization window sizes?
- What is the optimal number of disk nodes which can be connected to one client using two communication channels?
- How much do intelligent disk nodes improve overall performance when extracting, in parallel, relevant image information from image extents?
- What is the performance improvement when accessing image data from disk caches instead of accessing it from disks?
- How does an increased number of image-handling processors improve overall image access times?

To find answers to these questions, a discrete event simulation program was written in *Mathematica*. The simulation takes into account the time needed to compute extent distribution requests to disk nodes, communication time between client and disk node processors, SCSI disk block access times, two-dimensional block copies for extracting visualized image parts from extents, transfer of resulting

image parts through two communication links, receipt and assembly of image parts by the client processor.

Simulation parameters assume that processors are T800 transputers [3], that the channel throughput of a T800-based system is identical to the effective throughput of T800 transputer communication links (~1.6 megabytes per second) [7] and that disks have a raw transfer rate of 2.4 megabytes per second, a track to track access time of four ms and an average rotation time of 8.3 ms. Since multiple extents of the same image located on the same disk are generally stored on adjacent disk locations, we assume that mean seek time (fifteen ms) and average rotational delay are only applied to the first extent of an image on every disk. For consecutive extents of the same image, only track to track head displacement and transfer time are taken into account. This simulation setup has been validated by comparison with measurements carried out using a prototype T800-based image server composed of four image-handling processors, eight disk node processors and eight disks. The simulation has been extended to T9000 transputers which offer twice as much memory transfer throughput, ten megabytes per second data transfer bandwidth on their communication links and a crossbar switch for high-speed packet routing [5].

Simulation Results

Simulations have been carried out for different client window sizes (512x512, 256x256, 128x128, 64x64, 32x32 pixels), for different initial extent sizes (512x512, 256x256, 128x128, 64x64, 32x32 and 16x16 pixels), for between one and sixteen interconnected disk nodes and for full color images (three bytes per pixel). Image size is much larger than visualization window size (for example 2200x2200 pixels).

These simulations show that the optimal number of disk nodes associated with one client processor depends, to a large extent, on the number of communication links between that processor and the disk nodes. With a communication bandwidth limited by two transputer links, two disk nodes offer between 50% and 80% of asymptotic image access speed (image access speed obtained using a large number of disk nodes). Three or more disk nodes offer more disk access bandwidth but, due to the increased number of hops and the limited bandwidth of the two image handling processor's input channels, overall image access times do not decrease linearly (Figure 4).

Accessing large extents from disks reduces the time lost in head displacement at the expense of a higher latency (waiting time until full extent is available for transfer to client image processor). Small extents reduce latency time but require more disk head displacements and increased communication transfer overheads.

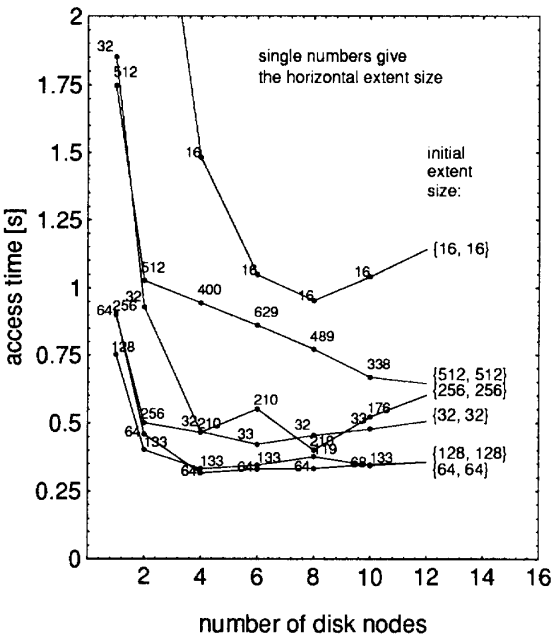

Figure 4. T800 single image handling processor MPMD access times for 512x512 visualization window.

The simulation shows that image subdivision into extents with sizes between ten and forty eight Kbytes (64x64 to 256x256 pixels) produces the best results (Figure 4). Global extent size is independent of the requested visualization window size. The extent partitioning algorithm described above computes the exact extent size to ensure that neighbors of a given extent are stored on different disks.

Parallel extraction of relevant image parts from extents at disk node level offers between 40% and 100% image access speed improvement for normal sized visualization windows (512x512 to 128x128 pixels). Each disk node includes a six megabyte cache memory. A list of cache resident extents specifies whether a given extent resides in cache or not. In the case of a cache hit, data is extracted directly from cache memory and sent through the communication channels to the client processor. For normal visualization window sizes, cache hits improve client access time by 5% to 30%. Image part extraction from cache brings speedup factors which are almost one order of magnitude higher than image part extraction or cache alone (Figure 5). Cumulated effects of cache hits and image part extraction provide speedup factors between 1.5 and four for normal visualization window sizes. For small visualization windows (64x64 to 32x42 pixels) speedup factors are between six and ten.

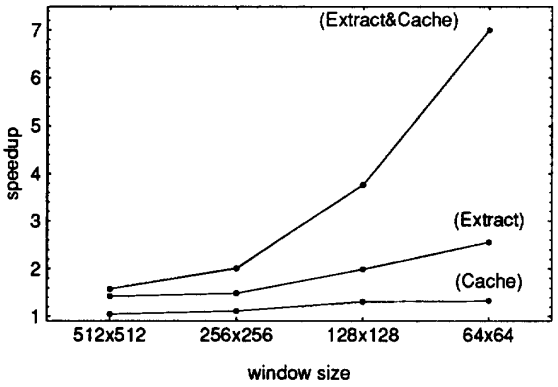

Figure 5. Speedup obtained by image part extraction, cache access and image part extraction from cache (single T800 image handling processor, eight disk nodes, extent size 128x128).

The T9000 transputer and its associated crossbar switch offer twice as much memory to memory transfer throughput and between six to ten times more communication bandwidth than the T800 transputer. For small visualization windows, simulations show that such an increase in communication bandwidth and processing power reduces image access times by a rather small factor (two to three). At large window sizes (for example 1432x1432 pixels) however, the T9000 transputer single image handling processor MPMD offers an image access throughput of eleven megabytes per second while, for the same configuration, the limited communication capabilities of the T800 transputer restrict its overall image access throughput to 2.7 megabytes per second (Figure 6).

Simulation also enables us to make a comparison between the present approach, where disk node processors extract relevant image parts from disk extents and transfer them to the image handling processors, and a general purpose RAID system, where one image handling processor receives complete extents from an array of parallel SCSI disks and extracts the relevant image parts. An ideal T800-based RAID system, with eight disks hooked onto eight independent SCSI channels with the same characteristics as above, is two to three times faster than a single image handling T800 processor connected to eight disk nodes (Figure 1a). This factor falls to 1.3 when the ideal RAID system is compared to an MPMD system consisting of four image handling T800 processors. Compared to a T9000 single image handling processor MPMD, an ideal T9000-based RAID is only 5% to 10% faster.

Real RAID systems based on a processor technology of the same generation as the T800 transputer do not offer their maximal bandwidth. System bus and memory band

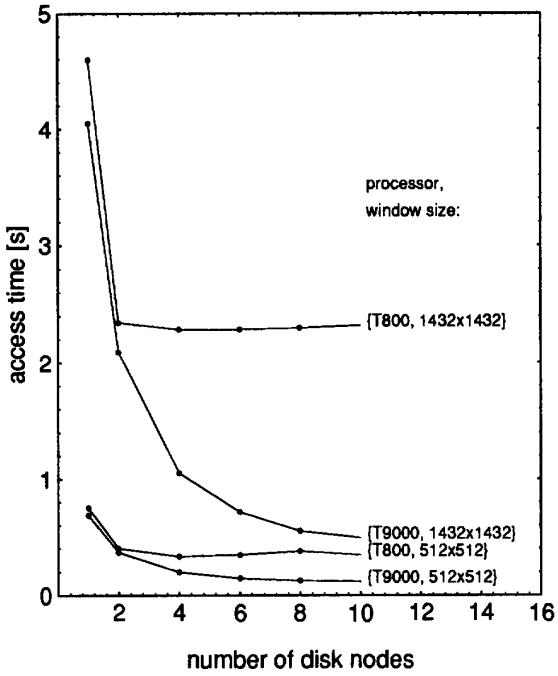

Figure 6. T800 and T9000 image access times for normal and large visualization windows (512x512 and 1432x1432).

width limitations reduce the available bandwidth by approximately a factor of two. File system overhead further reduces RAID bandwidth by a factor of three [6]. We can therefore assume that the presently described multiprocessor-multidisk architecture offers higher image access times than conventional RAID systems of the same technology. Moreover, thanks to its multiprocessor architecture, it offers significant image handling capabilities (zooming, adaptation of gray levels, iconization, creation of color maps, etc..) which are required for presenting images at the client's workstation.

Conclusions

Browsing through large pixmap images requires the image to be segmented into rectangular facets, which are fetched from disks or cache on demand. Optimal facet size depends on image size, on visualization requirements and on the available number of disk nodes. A server architecture consisting of one image handling processor and of eight disk nodes (Figure 1a) provides a cost-effective image server architecture. A 512x512 size window of a large-sized full color 24 bits per pixel image stored on eight disks in extent pieces 64x64 pixels can be fetched, transferred and received by a single image handling processor in 1/3 of a second. With four image handling processors connected to eight disk nodes (Figure

1b), image access time is reduced to 1/7 of a second. On a T9000 MPMD system with a single image handling processor, image access time is reduced to 0.115 second.

A higher throughput can be obtained when accessing full size images. For example, 768x512x3 color images (1.18 megabytes) can be accessed by a single T800 image-handling processor and two disks at the rate of two images per second (2.3 megabytes per second) and by four T800 image-handling processors and eight disks at a rate of 4.5 images per second (5.3 megabytes per second). With a single T9000 image-handling processor and eight disks one may access images at a rate close to seven images per second (8.14 megabytes per second).

Such performance figures are sufficient for building image servers offering fast access to large sets of pixmap images. Increased parallelization is worth while for large visualization windows (for example 1024x1024 pixels), as long as enough communication bandwidth between the MPMD processors is provided. Due to the disk access overhead for accessing small pieces of information, the performances at normal window sizes cannot be increased by applying a higher degree of parallelization. However, disk node parallelism offers strong potential for serving multiple requests simultaneously.

Simulations show that disk performance is a bottleneck when accessing images at normal and small visualization window size (256x256 pixels). At larger visualization window sizes, either the communication bandwidth between the processors or memory-to-memory block copy within the image handling processor becomes the bottleneck. Using an increased number of image handling processors directly connected to the network may provide a solution for eliminating the bottleneck of a single processor's limited communication and block copy capacity. However, such a solution depends on available network protocols and throughputs, and must be the subject of a separate study.

Acknowledgments

The author would like to acknowledge the contributions made B. Krummenacher, L. Landron et B. Tonelli who developed the hardware and the software of the T800-based MPMD prototype system.

References

[1] D. A. Patterson, G. Gibson, R. H. Katz, "A Case for Inexpensive Disks (RAID)", Proceedings of the ACM SIGMOD Conf. 1988, pp. 109-116.

[2] P. A. Chen, D.A. Patterson, "Maximizing Performances in a Striped Disk Array", Proceedings IEEE International Symposium on Computer Architecture, Seattle, 1990, 322-331.

[3] M. Homewood et al., "The IMS T800 Transputer", IEEE Micro, Vol. 7, No 5, Oct. 1987, 10-26.

[4] J. Wilkes, "DataMesh, Parallel Storage Systems for the 1990's", 11th IEEE Symposium on Mass Storage Systems, Monterey, Calif., Digest of Papers, IEEE Computer Press, 1991, 131-136.

[5] The T9000 transputer products overview manual, INMOS, 1991.

[6] A. L. Chervenak, R. H. Katz, "Performance of a Disk Array Prototype", Proceedings SIGMETRICS, May 1991, 188-197.

[7] B. Chardonnens, R. D. Hersch, O. Kolbl, "Transputer based distributed cartographic image processing", *Proceedings Joint Conference on Vector and Parallel Processing*: VAPP IV, Compar 90, (H. Burkhart, Ed), ETHZ, September 1990, Springer Verlag, LNCS 457, 336-346.