Implementation of Parallel Volume Visualization on the VC-1

2L−5

Yukio Sakagawa†, Satoshi Nishimura‡and Tosiyasu L. Kunii†
†The University of Tokyo
‡Kubota Computer Inc.

Abstract

The demand for faster image processing is increasing in such a way that it makes parallel processing necessary.

There is a great variety of computer graphic algorithms which can be utilized in parallel processor environments. In this work, we use volume visualization algorithms, and implement them on the VC-1, a loosely-coupled array of general-purpose processors with a conflict-free multiport frame buffer, being developed at the Kunii laboratory of the Computer Science Department, The University of Tokyo.

Distribution of the tasks and the locality of the informations among the processors will be the main issue of this study, while at the same time we evaluate the performance of this new architecture.

1 Introduction

Many scientific fields produce 3-dimensional array of data composed of many two-dimensional ones when analyzing volumetric objects, as in the Computed Tomography. It is required some training as it is difficult to visualize the three-dimensional structure by just looking at the several two-dimensional pieces (slices) of information. Taking medical application as an example, rendering the surfaces in a computer aids physicians to have a better idea of what is inside the body of patients, and thus can facilitate more accurate diagnosis.

There are several algorithms which have been used for the visualization of three dimensional data. Our objective is to take some of these algorithms and implement them on a multiprocessor environment, observing and taking as much advantage as possible of the characteristics of the machine architecture being used.

The algorithms used here are the Marching Cubes , volume rendering and slice planes method, to recreate and visualize the surfaces of interest from 3D medical data. In the near future, we plan to study other related algorithms.

The multiprocessor environment is the VC-1, a loosely-coupled array of general-purpose processors with a conflict-free multiport frame buffer.

2 The VC-1

VC-1 [6] is a loosely-coupled multiprocessor with a novel frame buffer subsystem called Conflict-Free Multiport Frame Buffer (CFMFB) which enables every processor to write any region of the screen without access conflicts. Fig. 1 illustrates the overall organization of the VC-1.

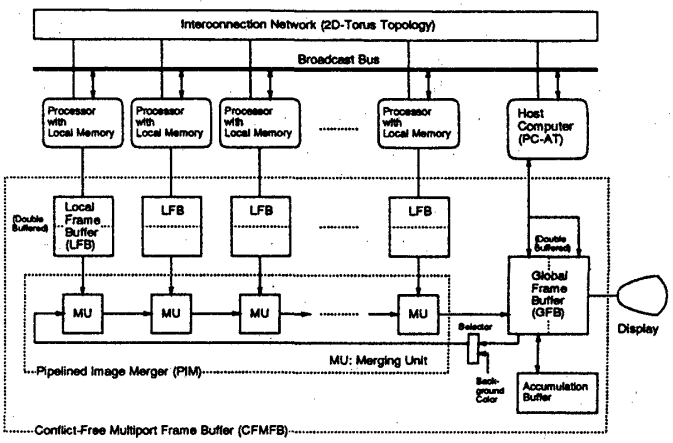

Figure 1: VC-1

Each processor element contains the Intel i860 at 40MHz with 8-Mbyte of local memory. The processors are interconnected in 2D-torus topology by point-to-point communication links with bandwidth of 2 Mbytes/sec each.

The CFMFB consists of *local frame buffers* (LFB's), a *pipelined image merger* (PIM), and a *global frame buffer* (GFB). The LFB exists for each processor and holds the sub-image (including Z-values) created by the corresponding processor. By using a method similar to the virtual memory technique, the LFB virtually holds the pixel information of the entire screen. The PIM periodically superimposes the sub-images stored in the LFB's and transfers the merged picture to the GFB. There is no communication overhead, as the transference is done automatically by the hardware. The Z-values are taken into account when the images are merged. The GFB holds the pixel information (including Z-values) of the entire screen.

The GFB is also provided with an accumulation buffer [3] for anti-aliasing.

3 Algorithms Description

3.1 Marching Cubes

The Marching cubes method [4] tries to estimate how the surface of interest intersects a cube formed by 8 neighbor points, where a group of 4 points is taken from one slice plane and other group of 4 points is taken from the neighbor slice, analyzing the density of the cube vertices.

There are 256 ways a surface can intersect a cube. However, considering its symmetry these 256 cases reduces to 14 cases, which can be easily pre-calculated and used in a look up table during the execution of the algorithm. Note that the cube can be intersected by more than one surface at a time. Once the intersection points are determined, one or more triangle planes can be drawn with each having its corresponding surface normal, which will be used later for shading the image.

3.2 Volume Rendering

Volume rendering [2] is a term to represent the process of showing a volumetric image without using an intermediate model, i.e., the volumetric image is obtained directly from the volume data. There are several algorithms for volume rendering, most of them derived from ray casting. The following is suitable for the hardware we use, due to PIM.

Let's consider a volumetric data, where the points of this data are represented by the pair density C (or color) value and the degree of translucency, α, where $\alpha = 0$ means a complete transparent and $\alpha = 1$ means a complete opaque material.

For a given viewline that crosses several elements, the observer would have the following view:

$V = C_0 + \alpha_0(C_1 + \alpha_1(C_2 + \alpha_2(C_3 + \alpha_3(C_4 + \ldots))))$

A higher index indicates the element is in a deeper position in the scene. This operation can be executed in hardware in the VC-1, through the PIM and the accumulation buffer.

3.3 Slice Planes

Once the volume is processed and defined, it is often desirable to remove sections or certain regions of the volume for visualization of its internal parts. The Slice planes method is one of the methods of volume visualization. The removed volume can be simple geometric shapes, defined by planes or halfplanes, or regions computed from other volumes.

4 Paralellization of Algorithms

In order to reconstruct surfaces of interest from the 3-dimensional data, we can divide the work in two phases: the processing of the slices (2-dimensional data pieces) and the processing of the images to be shown.

There are two ways to distribute the work among the processors: data parallel or function parallel. In the VC-1 the processors are homogeneous, so there is no reason to assign a specific task to any of them. We intend to work mainly over the data parallel approach, considering that all the processors will execute the same program codes.

The way data is distributed over the processors is important because it will define the load balancing of the system.

There are several ways data can be distributed among the processors: line distribution, block distribution, interleaved distribution, etc. In a generated image, normally there is not a uniform distribution of the objects on the screen. There are parts which require a few or no computation to obtain the image, while there are others that require heavy computation to be done. The best load distribution could be done subdividing finely the volume (interleaving)[1, 5]. However, this would increase communication among the processor and duplicate informations unnecessarily, because volume data locating at block boundary is duplicated to more than one processor.

There are no restrictions for the division of the image among the processors in the VC-1, as they have an LFB that can hold pieces of the image located anywhere in the whole screen. As the LFB holds up to 1/4 of the total screen frame precaution is necessary to avoid overflow, but this situation is easily solved, with swap of the LFB and download of the LFB to the GFB.

5 Summary

The VC-1 is being developed in our laboratory and it is not yet ready to be used. We can just speculate about its expected performance. Here we just described in general lines how we intend to proceed to parallelize algorithms on the VC-1. The architecture of the VC-1 does not impose any predetermined distribution of the tasks or the data among the processors, making the distribution very flexible.

References

[1] B. Corrie and P. Mackerras. Parallel volume rendering and data coherence on the fujitsu ap1000. Technical Report TR-CS-92-11, The Australian National University, August 1992.

[2] R.A. Drebin, L. Carpenter, and P. Hanrahan. Volume rendering. *ACM Computer Graphics*, 22(4):65–74, August 1988.

[3] P. Haeberli and K. Akeley. The accumulation buffer: Hardware support for high-quality rendering. *ACM Computer Graphics*, 24(4):309–318, August 1990.

[4] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM Computer Graphics*, 21(4):163–169, July 1987.

[5] P. Mackerras. A fast parallel marching-cubes implementation on the fujitsu ap1000. Technical Report TR-CS-92-10, The Australian National University, August 1992.

[6] S. Nishimura, R. Mukai, and T.L. Kunii. A loosely-coupled parallel graphics architecture based on a conflict-free multiport frame buffer. In *Proceedings of the Third Workshop on Future Trends of Distributed Computing Systems*, pages 411–418, Taipei, Taiwan, April 1992.