A Distributed Image Processing Environment VIOS III and it's Performance Evaluation

Hiroshi Matsuo, Kosaku Nakada and Akira Iwata
Nagoya Institute of Technology, Dept. of Electrical and Computer Eng.
Gokiso, Showa, Nagoya 466-8555, JAPAN
matsuo@elcom.nitech.ac.jp

Abstract

We proposed a distributed image processing enviroment VIOS. In this paper, the third version, VIOS III is proposed. In VIOS III, a new parallel processing language VPE-p which has flexible syntax for describing parallel algorithms has been developed. And the new programmable buffer for accessing global variable through local area network is also proposed. The description ability for parallel image processing algorithms and processing performance using workstation clusters and multi processor system are investigated by several image processing and recognition algorithms.

1. Introduction

There are a lot of high-end computers available today. However, even using those computes, image processing and image recognition algorithms still demand enormous computing time. Now a days, it is easy to get many workstations in one research group. For this reason, we proposed a distributed image processing environment VIOS for parallel image processing. This software environment uses many workstations connected by local area network or local bus at the same time.

On the other hand, multi CPU system and workstation cluster system have become commonplace. While using these systems, knowledges concerning the system programming likes network programming and inter process communication program (IPC) are needed. Then PVM[2] and MPI[3] are proposed as a library to construct the network cluster environment. Network and IPC knowledge are still needed because of these libraries are implemented using low-level routine.

The image processing environment such as XITE[5] and Khoros[4] are developed, but these programming environment can't describe parallel programming which considers data parallel of image processing algorithms.

VIOS[1] is a general-purpose distributed and parallel image processing environment which has the parallel image processing language named VPE-p that can be described parallel programming easily. But the ability for the parallel description of VPE-p which developed with VIOS II[1] is low because the parallel processing unit is only based upon pixel, row (column) and block.

In this paper, we proposed new distributed image processing environment VIOS III. VIOS III have the following features,

- New parallel image processing language VPE-p which provides flexible parallel processing description ability
- The programmable global variable buffer by which users can describe the method for integration of each buffers which is located on each workstation.

And the description ability and the processing performance of a new language VPE-p are evaluated by describing various image processing and recognition algorithms.

2. A distributed image processing environment VIOS

2.1. The architecture of VIOS

The VIOS system is composed of three kind of processes named IPU, OM and VPE. These process run on different workstations (or different CPU on multi processor machine) and are connected by local area network (or local bus). VPE(Visual Programming Editor) is a user interface process for VIOS. VPE has a programming language for image processing named VPE-p. Using this language, new image processing

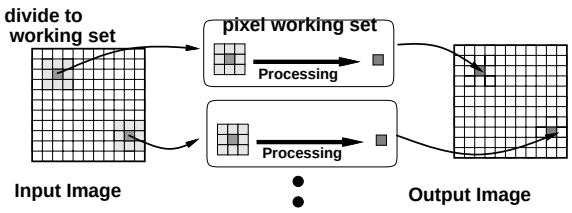

Figure 1. The parallel image processing working set

module can be described. IPU(Image Processing Unit) is implemented as a virtual machine, which execute instructions from OM. This instruction is a stack-based language like FORTH. Many IPU can be located on a local area network. OM (Object Manager) is the management process of image data, IPU and scheduling of module execution.

We proposed the new type of image division data structure for distributed and parallel processing called the "parallel image processing working set" which is shown at Figure.1 (hereafter called it working set). The working set has three data as follows (1) dependence of the data for data parallel execution (2) A pixel or pixel set named the main pixel which can read and write from the data parallel execution sequence (3) "The surrounding pixels" that are only referred from image processing modules.

A new image processing algorithm is described by VPE-p, which has C like syntax. A type of parallel execution is described using the type of working set. Four type of the working set is implemented as follows.

Pixel parallel: Each output pixel value can be calculated independently from other output pixel value. (Ex. Image subtraction, Laplacian filter etc.)

Row (column) parallel: Each output row or column value can be calculated independently from other output row or column value. (Ex. Matrix multiplies, one dimensional Fourier transformation etc.)

Block parallel: Each output block value can be calculated independently from other output block value. (Ex. Image compression etc.)

Non parallel: Parallel execution is impossible.

2.2. Extension in VIOS III

Easy description of parallel image processing becomes possible by using the working set data structure when the image processing algorithm has data parallel feature clearly such as Laplacian filter. However some operations (for instance the procedure to find maximum value) which seemed to be parallelized easily can not be described in parallel using VPE-p on VIOS II. Because the description which uses the result of distributed data from each working set is impossible in VPE-p.

To enable more flexible parallel image processing algorithm described, VIOS III and its description language VPE-p add the following extensions

The serial part is introducebd in the module: To describe serial part in module, the module is divided into two part, one is parallel processing description part where the algorithms apply to each working set is described. The other is serial processing description part where the algorithms can not be described parallel. Combining this extension with the working set, parallel description ability greatly improved

Network transparent global value and pixel data : Network transparent global value that can be accessed through network is introduced. This variable can be shared between working sets, which is located on the different workstations. Furthermore, the references of pixel value extend to surrounding pixels is also enabled.

Programmable global variable buffer: In the image processing algorithms like the Hough transform or the geometric hashing, strict exclusion procedure is not needed for accessing the global variable. Accessing to a global variable can be replaced with the access to the local variable, which is integrated by using appropriate data integration rule that is executed when the execution step of the module is finished. Then a programmable global variable buffer for the distributed image-processing environment VIOS is proposed.

3. Parallel image processing language VPE-p

VPE-p consists of two parts, one is the module description part which describes image processing algorithms, and the other is main flow description part which describes the relation between each module.

3.1. Extension of parallel description within image processing module

The module definition part is a part where an actual image processing is described. The type of the working

```
                                    Module definition part
module Max(a:input, x:output)
int a on box[32][32];               Argument declaration part
int x;
{
 int max[16][16];                   Grobal valiable declaration
 int i,j;                           part
 parallel{
  int xx=hotx(a)/32;
  int yy=hoty(a)/32;                Parallel processing
  int i,j;                          description part
  max[xx][yy]=a[0][0];
   for (i = 1; i < 32; i++)
     for (j = 1; j < 32; j++)
       if (max[xx][yy] < a[i][j])
         max[xx][yy] = a[i][j];
 }
 x = max[0][0];
 for (i = 0; i < 16; i++)
  for (j = 0; j < 16; j++)
    x = (x > max[i][j]) ? x:max[i][j];
}                                   Sequential processing
                                    description part
```

Figure 2. The example of module (maximum value detection)

set of input and output parameters are specified with the variable name and image-processing algorithm is also specified. The input image data name is specified ahead of reserved word "input", the output image data name is specified ahead of "output" and variables are specified ahead of "parameter" in the module declaration part.

> module *module_name*(*input_image*,:input, *output_image*,:output,*variables*,:parameter)

In the argument declaration part following the module declaration part, the types of input and output parameter are specified as follows.

> data_type *variable_name* [on *working_set_type* [cache *n*]]

When argument is Image type[1], the working set type is specified following reserved word "on". The width of the surrounding pixel is specified following "cache" when the surrounding pixels are needed. A module consists of two parts, one is the parallel description part and the other is serial description. Figure.2 shows an example of describing the maximum value detection module using "box" working set.

3.2. Access to the remote variable

In VIOS II, it was impossible to access variables outside the working set. However, this limitation became a serious demerit of describing various distributed image-processing algorithms. Then, to access the pixel data that is outside of the surrounding pixels of the working set transparently, VPE-p has been extended in VIOS III. But an enormous access cost is needed to access the pixel data outside the working set because network latency is large and exclusion control is also needed.

On the other hand, in some image processing algorithms, the accuracy of pixel value is not needed. Then we proposed five policies as accessing the pixel value outside the working set. Users can select these access following policies explicitly, (1) The access outside the working set is not checked (no check:default), (2) A true value is acquired though the network (get), (3) Substituted by pixel value in the working set (near), (4) Interpolated using the working set (complete), (5) The specified constant is substituted (const n).

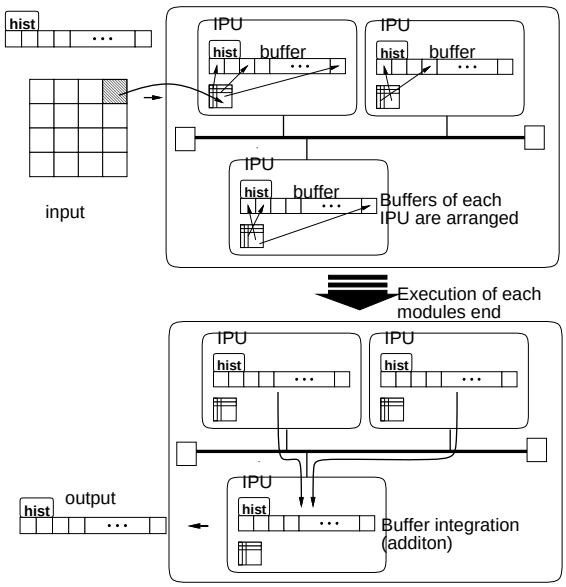

Figure 3. Histogram calculation using programmable global variable buffer

3.3. Programmable global variable buffer

Access to a global variable becomes the bottleneck of the system in the distributed processing environment with a large latency of the communication between each processor. However many image processing and image recognition algorithm are executed treating a global variable as a local variable, and each variable can be integrated when the execution of modules ends. In VIOS III, the buffer of each module that described such characteristics of the image processing, the user

[1]The Image type was introduced as a new data structure to the image in VIOS. This type has various information(each pixel type of image, size of image and history of processing).

```
module Histogram(in:input,histogram:output)
int   in    on box[32][32];
int   histogram[];
#vios mutex histogram off add
{   parallel
   {  int x,y;
      for (x = 0; x < 32; x++)
         for (y = 0; y < 32; y++)
            histogram[in[x][y]]++;
   }
}
```

Figure 4. The histogram calculation module

specified an integration procedure of each buffer and the speed-up of the access to a global variable was achieved.

Figure 3 shows the flow of the data when the buffer integration process is used to calculate the histogram of the density. Each IPU is accessed in buffered global variable "hist". Each buffer is integrated into one when all the execution of the parallel processing description part ends.

Maximum, Minimum and Addition are implemented as an integrated method of the buffer now. The control method of the buffer integration is described following the exclusion control method to a global variable. The control method of the exclusion and the method of the buffer integration are described following the variable name.

4. Performance evaluation of VIOS III

4.1. Performance evaluation by multiprocessor system

The effect of the number of IPU which execute image processing module were evaluated using Sun Microsystems SS-1000 (OS is Solaris 2.5.1 and 128MB main memories), which has 8 CPU connected to local bus. Parallel image recognition program including Sobel filter, thinning, binarization using Otsu method, Hough transformation is implemented using extended VPE-p for this experiment. The program used for this experiment is appended to the appendix. The input image consists of 512x512 pixels and 8259 voting point (about 3 % of the number of all pixels) and 180x1446 pixels voting space are used for Hough transformation.

Performance progression is shown in Table 1. As the number of IPU increases, performance also increases. Execution time is 1.8 times faster when running 2 IPU, 2.9 times faster when running 4 IPU and 4.0 times faster when running 8 IPU on SS1000. A decrease at execution speed when IPU increases is due to the start time of modules and overhead for division and integration time of images. In addition, to compare the execution speed between the program which is described by C language and described by IPU-p. As a result, the effect of performance decrease when describing with IPU-p is about 20 % when describing in C language. This is because VIOS generate general parallel code if only 1 CPU is used.

4.2. Performance evaluation by workstation cluster

The performance evaluation on the workstation clusters (OS: Solaris 2.5.1, Memory: 128MB, CPU: PentimuPro 200Mhz) which connected to 100 Base TX switched network using switching HUB (Bay networks model 28115). Same program is programmed using PVM 3.3 and C. The result is shown in Table 2. When only 1 CPU is used, the effect of performance decrease when describing with IPU-p is about 20 % when describing same program in C language, 5 % in PVM.

The number of CPU increased, good performance advancement is achieved using VIOS III and VPE-p compared with PVM and C. The reason of this result is that parallel Hough transformation need to transfer a lot of table data, network modules of VIOS is implemented using thread which executed parallel.

4.3. Performance evaluation of programmable global variable buffer

Effectiveness of the programmable global variable buffer was examined by using the program, which generate the density histogram from the input image show in Figure 4. The result shows in Table 3. The histogram calculation time in the Table 3 indicates the real processing time to generate the histogram actually on each IPU and access cost indicates the access cost to the global variable though the network. As shown in the table, if there is no histogram integration, the enormous access time to the histogram array though the network are needed. On the other hand, only integration was performed after the histogram generation ended was performed. As a result, large improvement of processing time was obtained.

5. Conclusion

Distributed image processing environment VIOS III which uses workstation cluster or multiprocessor machine is proposed. This system is composed of three kinds of processes VPE, OM and IPU. It was confirmed

Module name	Description Language				
	C	VPE-p			
	Number of CPU				
	1	1	2	4	8
Sobel filter	0.90	1.45(1.0)	0.87(1.7)	0.44(3.3)	0.28(5.2)
Thinning	0.29	0.54(1.0)	0.33(1.6)	0.24(2.3)	0.17(3.2)
OTSU	0.15	0.61(1.0)	0.42(1.5)	0.32(1.9)	0.27(2.3)
Hough trans.	7.93	10.99(1.0)	5.44(1.8)	3.06(3.3)	2.60(3.9)
local max detect	4.68	4.69(1.0)	2.52(1.9)	1.84(2.6)	1.03(4.6)
max detect	0.12	0.28(1.0)	0.20(1.8)	0.15(2.9)	0.11(4.0)
sum	14.07	17.57(1.0)	9.78(1.8)	6.05(2.9)	4.46(4.0)

Table 1. Performance for pattern recognition on Sparc Center 1000 (unit: sec)

Module name	Description Language						
	C	VIOS III and VPE-P			PVM and C		
	Number of CPU						
	1	1	2	4	1	2	4
Sobel	0.69	0.78(1.0)	0.47(1.7)	0.29(2.7)	0.73(1.0)	0.38(1.9)	0.20(3.7)
Thinning	0.09	0.15(1.0)	0.21(0.71)	0.29(0.52)	0.14(1.0)	0.08(1.8)	0.06(3.5)
Otsu binarization	0.06	0.42(1.0)	0.56(0.75)	0.67(0.63)	0.10(1.0)	0.06(1.7)	0.05(2.0)
Hough Trans.	6.44	6.97(1.0)	3.60(1.9)	2.10(3.3)	7.44(1.0)	6.22(1.2)	6.58(1.1)
Local Max detect.	2.05	2.86(1.0)	1.53(1.9)	0.82(3.5)	2.21(1.0)	1.42(1.6)	0.81(2.7)
Max detect	0.05	0.13(1.0)	0.15(0.87)	0.26(0.50)	0.06(1.0)	0.04(1.5)	0.03(2.0)
Sum	9.38	11.31(1.0)	6.52(1.7)	4.43(2.6)	10.68(1.0)	8.20(1.3)	7.73(1.4)

Table 2. Performance for pattern recognition on Workstation Cluster (unit:sec)

	Without buffer integration	With buffer integration
Histogram calc.	1.98 sec	1.98 sec
Access cost	27442.62 sec	0.30 sec
Sum	27444.60(1.0)	2.28(12037.11)

Table 3. Performance comparison of explicit buffer integration algorithm

to be able to describe various parallel image processing algorithms using parallel image processing description language VPE-p and evaluated the performance. Moreover, the programmable global variable buffer was proposed and implemented. And the effectiveness of this proposed method was shown by the experiment. This system has no special hardware, only uses workstations connected by local area network. It becomes possible to use many workstation and multiprocessor workstation effectively.

References

[1] H.Matsuo and A.Iwata. A distributed image processing environment VIOS II. ACCV93:715-718 1993.

[2] A. Geist et al. Parallel Virtual Machine A Users Guide and Tutorial for Networked Parallel Computing. MIT Press 1994.

[3] C. H. Still. Portable Parallel Computing Via the MPI1 Message-Passing Standard. Computers in Physics, 8(5):533-538, 1994.

[4] Konstantinedes and Rasure. The Khoros Software Development Environment for Image and Signal Processing, IEEE Trans. on Image Processing, 3(3):243-252, March 1994.

[5] Otto Milvang, Tor Lnestad. An Object Oriented Image Display System, Proc. of 11th ICPR, D:218-221, Oct 1992.