7D−3

An Implementation of FIGURE-1 Processor for Heterogeneous Computer Networks

Weda Nurdayat* Takashi Yamanoue* Takeshi Fujiki*
Hiroyuki Anzai+
* Kyushu Institute of Technology + Kyushu Kyoritsu University

August, 1993

1 Introduction

A heterogeneous computer networks can be viewed as a set of special purpose computers. A parallel program for a heterogeneous computer network has ability to increase efficency of the computing.

We try to implement a processor of the parallel programming language FIGURE-1 on a heterogeneous computer network.

In section 2, introduction of parallel programming FIGURE-1 is shown. In section 3, an implementation of FIGURE-1 processor for a heterogeneous computer network is shown.

2 FIGURE-1

Parallel programming can be exploit such as object oriented programming language C++. In the object oriented programming language, a program is represented by a message passing between objects, and message passing between objects is analogous to the communication between nodes of parallel computer.

However, most of the previous programming languages describe only one-dimentional notations, so that difficult to grasp image of a message passing between process. FIGURE-1 is a two-dimensional object based programming language.

In FIGURE-1, object(process) are represented by rectangles, and a message passing(or data flows) between object are represented by arrows. An object is represented by a retangle and communication between object is represented by an arrow. These figures include such character as |,-, +,<,>,V ,and ^. Therefore, such conventional editor on normal character displays as emacs and vi can be used to write FIGURE-1.

A FIGURE-1 processor is a tranlator which translates the the program, written in figure, into a set of C program. Each process represented by rectangle is transformed into one program, and each message passing represented by an arrow is transformed into I/O statement of programs. This processor exploits several techniques to solve syntactic recognition, and it is generated by compiler-compiler MYLANG.

FIGURE-1 consists of a fixed number of object and message passing between each other. The list of nodes(hosts) and common definitions of the program proceed the figure of object and message passing.

```
hosts[]=
{{"object_name_0", "host_name_0"},
{ ..............  , .............,},
{"object_name_n-1", "host_name_n-1"} };
```

An object is presented by a rectangle which contains the list of functions, the list of variables, the list of initial operations, and the list of operations.

```
+----------------------------------------+
|  object object_name                    |
|   functions                            |
|     [ function definition in C ]       |
|   var                                  |
|     [ variables ]                      |
|   operations                           |
|     list_of_input_message              |
|     >> list_of_output_message          |
|     [ statement in C ]                 |
+----------------------------------------+
```

In order to send and receive messages, the following functions are available in the list of operation.

put_message(*output_message_name, address_of_variable*)

Send the message which is the content of variable.

get_message(*input_message_name, address_of_variable*)

Receive the message and assign its value to the variable. This function waits for the message until it comes.

get_message_nw(*input_message_name, address_of_variable*)

If there is a message in the internal message queue, receive the message, assign its value to the variable and return 1. If there isn't message in the queue, return 0 without waiting.

3 FIGURE-1 Processor for Heterogeneous Computer Networks

The FIGURE-1 processor for heterogeneous computers networks generates a makefile, a startup file and C programs from a FIGURE-1 program and a system configuration file.

The system configuration file includes the following informations.

1. The way to compile an object and the way to startup an object for each computer.
2. The way to communicate between objects for each pair of computers.

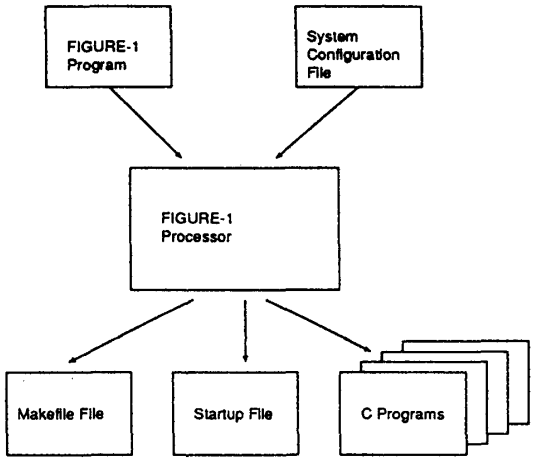

Figure 1: Diagram FIGURE-1 Processor

For an example, a parallel program which displays mandelbrot set on the heterogeneous computer network like Figure.3, is transformed into C programs as follows.

1. m_dealer.c, which deals out areas to m_consumer.c, runs on SRM of IPSC/i860.
2. m_consumer.c, which compute the area that is sent from m_dealer.c and send a result to m_collector.c, runs on 4 node of IPSC/i860. Communication between the m_dealer.c and a m_consumer.c is performed by functions such as **csend** and **crecv** of the IPSC.
3. m_collector.c, which collects the results of m_consumer.c, runs on Sun/SPARC-2. Communication between a m_consumer.c and the m_collector.c is performed by function such as **read** and **write** of TCP/IP. The result are sent to m_display.
4. m_display.c, which displays the results, runs on Sun/SPARC-2. Communication between a m_collector.c and the m_display.c is performed by functions such as **read** and **write** of TCP/IP.

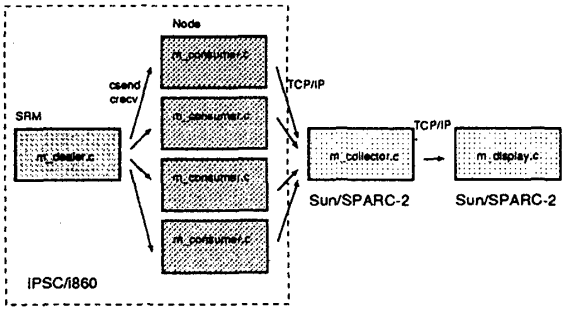

Figure 2: A Mandelbrot programs runs on a Heterogeneous Computer

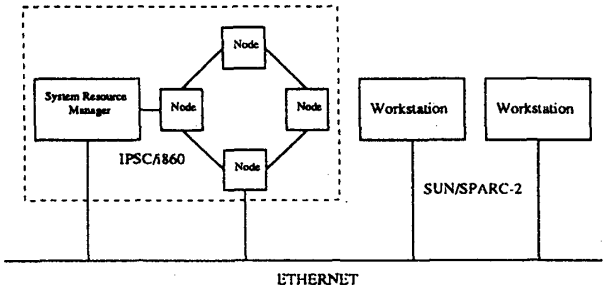

Figure 3: An Architecture of Heterogenous Computer Networks

4 Conclusion

FIGURE-1 helps programmer grasp the image of parallel programs more easly, and its processor can help programmers develop program faster for heterogenous computer networks. FIGURE-1 processor can be used for more complexity heterogenous computer networks by modified message passing function.

References

[1] P.J.Hatcher,M.J.Quinn,R.J.Anderson, A.J.Lapadula, B.K.Seevers,and A.F.Bennett, *Architecture-Independent Scientific Programming in Dataparallel C: Three Case Studies* ,Proceeding Supercomputing '91,1991,pp.208-1991.

[2] T.Yamanoue, H.Hayata, H.Anzai, *"A Figure Programming Language for Parallel Supercomputers"*, Transputers and Parallel Application, IOS Press, pp.209-214(1993).

[3] T.Yamanoue, H.Hayata, H.Anzai. *"A figure langauge for distributed system descriptions and its pre-compiler which can parse figures "* ,Proceeding ICSC'92:Second International Computer Science Conference, pp.425-431(1992).