A Parallel and Distributed Environment in GA.SP Data Acquisition System

B. D'Avanzo. M.De Poli, G. Maron, M.L.Mazza, S. Scanferlato, G.Staiano, X.N. Tang G. Vedovato

INFN-Laboratori Nazionali Di Legnaro
Via Romea 4, I 35020 Legnaro (Padova)
Italy

Abstract

A parallel and distributed environment of GA.SP data acquisition system is developed in INFN-LNL. The system consists of various tools distributed on from UNIX workstations to Transputer NETwork(TNET) and implemented by Yacc, X-Window and Occam Toolset etc. In general, TNET is a multichain architecture in which each chain is a pipeline machine for number crunching. Parallel programming on this TNET is assisted by NEO compiler and TNET shell and communication between TNET and outside is through a VME workstation which is connected to other high level control workstations by Ethernet.

1 Introduction

GA.SP is an array of suppressed Germanium detectors which consists of 40 Ge-suppressed dectors(70-80% efficiency) and 80 BGO detectors(multiplicity filter) and its data acquisition system is designed to deal with 50 μs/event and 2MBytes/s. In order to achieve this high speed on-line data processing, a parallel and distributed architecture was designed and reported elsewhere[1]. The general layout of this data acquisition system is in Fig. 1 and major components of this system are summarised as the follows:

1) A special designed distributor[5] injects the event from FERA bus into the TNET based on round robin and interrupt request scheme.
2) About 100 nodes of Transputer NETwork - TNET is used as "number crunching" machine for the major computation in the data processing.
3) A number of workstations connected via a local network is used to analysis data and control data acquisition system, the latter is the host of TNET.
4) A Nuclear Experiment Oriented(NEO) Language has been developed to facilitate writing data analysis program.

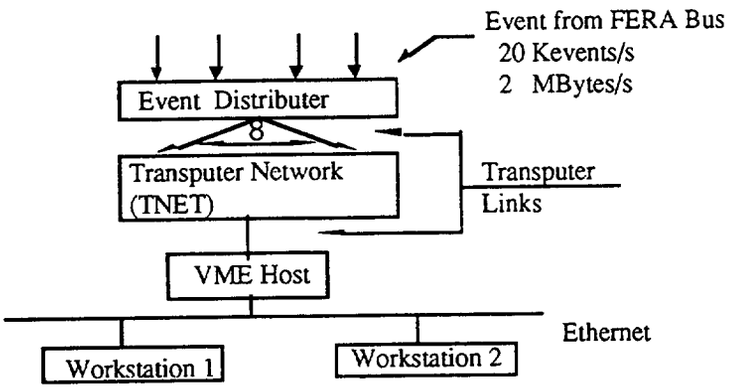

Fig. 1: General Layout of GA.SP Data Acquisition System

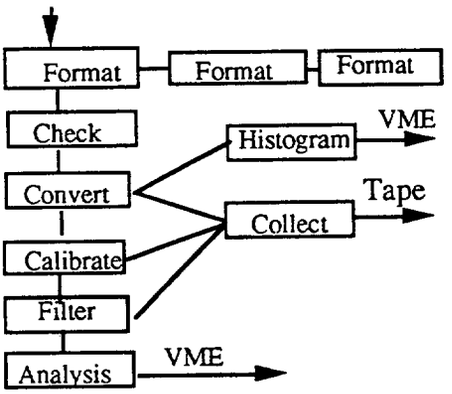

Fig. 2 A chain of Transputers

A great effort has been dedicated to developing the appropriate softwares to build an integrated parallel and distributed environment based on this architecture since 1991 and we report four major modules here: User Interface, Unix Server, NEO Compiler and TNET Shell.

2. User Interface

The aim of the User Interface is to hide all the underlying NEO, VME and TNET processes which are introduced in the following sections and to give users a comfortable and effective environment for nuclear measurements. The whole interface has been developed on an

HP 9000-750 series computer using X-Window 11R4 and Motif1.2 standard libraries. As a development tool, HP Architect 2.0 is also extensively used.

The main part of the interface is the "Open Control Panel" which handles the TNET commands(Start, Stop, etc.), the setting up of the whole system (VME, TNET and CAMAC), the Tape operations like mount and finally the execution of any predeclared display or analysis program.

A particular effort has been spent to realize the graphic display interface, "Spectra Manager", since we have to deal with a large number of monodimensional spectra either for measurement control or for on-line analysis. For this reason a dynamic memory based network data structure has been developed to handle the properties of the histograms[3]. The data structure does not impose any limitation to the maximum number of histograms handled and is used by a geometry manager for the layout of the display. Actually the user can switch between 16 different "screens" with any number of histograms displayed and for every screen he can recursively select histograms to obtain from them a new higher level display configuration. To allow easy comparisons, every histogram can be in a "independent" or "homogeneous" state with regard to scaling and offsetting operations which are performed dynamically with respectively private or global scroll bars. Standard analysis tools like energy and width calibration, peak search, peak integration, multiple gaussian fit, etc are also provided.

A two dimensional display interface with the same characteristics and an interactive NEO program generator tool are now under development.

3. Unix Servers

Because of the complexity of modern data acquisition system, one of widely used methods to build such system is the distributed computing technology. It consists of a network connecting the needed computing nodes in which each one performs some special functions which it is apt at. The Remote Procedure Call(RPC) technique is then used to deal with the communication among the nodes in the network. A VME Unix workstation is chosen as a host to the TNET with 4 x 64 MBytes histogram database which can be accessed by Transputer directly. Four types of server have been developed on this Unix workstation to support the communication between HP workstation and TNET .

The Host Iserver is a modified Inmos Toolset Iserver which is transplanted into VME workstation to support loading and debugging Occam program because the object code of NEO language is Occam. The original Iserver only supports the link level accessing Transputer and is not enough in our case. By using shared memory IPC mechanism, our host server can support both link and shared memory communication schemes. The current Host uses two Transputers to separate the compilation and communication in order to support concurrent activities of the running and compilation of NEO program.

The Message Server is in charge of all the information exchange between the VME workstation and TNET. Its communication is through both one dedicated Transputer link and a dual port memory in which the link passes the special signal to coordinate the read/write operation and the dual port memory contains the corresponding parameters whose dimensions are usually big. Because we adopt dual port memory mechanism, the message server actually consists of two parts, C and Occam components respectively. Both can look through the same external memory to exchange message.

The HP Server is facing the HP by receiving request from the HP workstations and calling the corresponding TNET servers to execute them. The main issue here is to divide the remote requests into different groups so that each group can use its appropriate protocol to pass parameters avoiding unnecessary data transfer. The reason is that RPC call always transmits the fixed amount of data and different requests usually don't have the same parameters. For example, the Start TNET hasn't any parameters to pass but the command of Loading program to TNET needs a large array to contain the object program. So, grouping remote requests will increase the RPC call efficiency.

The last server is the Histogram Server which dumps the histogram database to HP workstation for visualization. Due to the big dimension of histogram, RPC call is not enough to deal with this dumping and socket mechanism is adopted to access the VME histogram memory directly. During current operation, all workstations are connected by Ethernet with TCP/IP protocol which is only at the speed of 500KBytes/s. A FDDI ring has been set up and tested showing the speed of 2MBytes/s for the future experiment.

4. NEO Language

As the Occam is a concurrent language and unfamiliar to ordinary users, a Nuclear Experiment Oriented(NEO) Language has been developed to support easy programming on the TNET. Since the NEO has been reported in [2], we only summarize the characteristics of NEO here and introduce its new features in current version.

1) The NEO is a section based C-like structure language. There are six sections in the NEO: Format, Check, Calibration, Histogram, Filter and Analysis which are corresponding to the general steps used in data processing of Nuclear Physics Experiment. Each section is a independent computation unit and can be executed concurrently. The event schedule between sections is manipulated by the compiler and programmer need only concentrate on the logic part of his NEO program.

2) The NEO not only supports basic data types such as Integer, Boolean and Real but also high level data structures like histogram which is necessary in data processing of Nuclear Experiments. So, programming in NEO will become more easily by using those domain specific structures.

3). Each section of the NEO program is allocated to one or several Transputers depending on the its complexity, see Fig. 2 . But the assignment of Transputers is transparent to the user. Each section is accessed by its logic name no matter what actually the architecture is. The NEO hides end-users from the details of hardware architecture and assists them to do concurrent programming.

NEO is implemented by standard Unix tools Yacc and Lex. Thus, it's easy for us to add the needed features. The first version of NEO has no I/O abilities and the program behaviour can't be changed during the running. But interactive communication with NEO program is compulsory during the program debugging and experiment monitoring. A new data type MESSAGE is introduced in the NEO to assist user interaction with the running program dynamically.

There are three levels of MESSAGE which can be used to change program behaviour: single variable, array of variables and a section of program. A user can associate a BOOL variable with a procedure and control whether using that procedure or not by issuing the MESSAGE command to modify the value of the BOOL variable. For the calibration constants, user can send an array MESSAGE to adjust the corresponding calibration process. Furthermore, if user wants to dramatically replace his algorithm in one section, he can dynamically reload whole section program while keeping other sections unaffected. For example, dynamic loading the Analysis section doesn't affect the data flow to the Tape as well as the Filter section processing.

5. TNET Shell

TNET is the key part of GA.SP data acquisition system and we can look through it in two levels. On the first level, TNET is a multichain structure in which each chain is independent and works in pipeline, see Fig. 1. Because the event distributor has two T222 Transputers on board, upmost 8 chains can be supported. The strategy of distributing event is combining the round robin and interrupt request which is sent by each chain so that not only events can be divided evenly but also the unready chain can be skipped to achieve both high speed and fault-tolerance. On the second level, each chain consists of the fixed sections corresponding to NEO language and they are arranged according to the requirement of data analysis, see Fig. 2. Because we use

T222 Transputer for the Format and Check sections and T800 for others, the Convert node is inserted in between to transfer between 16 bits and 32 bits which is unseen from the NEO point of view. One of key principles in designing pipeline machine is to keep every stage balance. TNET solves this in two ways. First, for those sections whose computation are heavy as well as every event must be processed in them, the farmer Transputer array is used to speed up their computation like Format section. Second, for those chains in which events are not required to be processed in 100 percent, the software interrupt request is used to make it work as fast as possible so that the whole chain processing is not blocked, such as Histogram and Analysis sections.

The TNET shell is built on the above architecture and assists user to operate TNET. There are three modules in TNET shell: a special router for message passing, a command interpreter for data flow control, and the tape driver for storing data.

The TNET router is a distributed one because each Transputer uses its neighbourhood structure to direct message passing and the whole structure is represented distributively on each Transputer node. Thus, the complexity of routing function in each node depends on how many connections it is linked to its neighbours. From Fig. 2, we can see the router for the first Format is more complex then that of the second Format. Due to the limited links of each Transputer, the TNET router implements the virtual channel to overcome this limit so that different information flows such as events, commands, feedbacks, loaded program, partial and final results etc. can fly through TNET freely and safely through the four available links in each node. Because there is only one link to connect VME workstation and TNET, the router use a special propagation strategy to broadcast some commands in order to reduce the link overhead between VME host and TNET, especially when dynamically changing program.

The command interpreter of TNET just relies on this TNET router to direct information flows in the network and the main idea here is to reduce the number of control signals which are across the Transputer link because it will deteriorate the network performance. For example, when using the polling of interrupt request to avoid sending unacceptable message, it's better to use a simulated buffer on the output link rather than asking next Transputer through the link directly. The most difficulty command in the implementation point of view is Dynamic Loading command because it needs to kill some active process while keeping the event flow active. We implemented it as follows: first buffering event message, then sending process killing signal to stop current process and using lower level facility provided in Occam Toolset[4] to load program again. After the replacing operation, release the event flow so that the processing returns normal.

The tape driver is written in Occam and run on a SCSI board which can arrive up to the speed of 500KB/S for EXBytes-8500. The data path for collecting data is organized as a three level tree structure because of the link limitation of Transputer and two board controller are used to reach the speed of logging data to tape in 1MBytes/S. There are two modifications to the original tape driver, one is to support STK-4280 tape for higher speed and the other is to provide asynchronous tape commands for concurrent operations, such as writing, loading and rewinding.

Reference

[1]. D. Colombo et al, "The Transputer Based GA.SP data Acquisition System", IEEE Transaction on Nuclear Science, April 1992.
[2]. D. Colombo et al, "A Multiprocessor Based Language for Data Processing in Nuclear Physics Experiment", IEEE Real Time '91, Julich, Germany, June 24-28, 1991.
[3].D.E. Knuth, " The Art of Computer Programming", Vol. 1, Addison-Wesley Pub. Company, 1978.
[4]. Parsytec, "Occam 2 Toolset User Manual", Dec. 1990
[5]. Z. Cavedini et al, " A FERA to Transputer Interface board for the Gasp Experiment", LNL-INFN 58/92.