A transputer-based parallel expert diagnostic system

R. Campanini a, I. D’Antone a, G. Di Caro a and G. Giusti b

a Dipartimento di Fisica dell’Università degli Studi di Bologna and INFN Via Irnerio 46, 40126 Bologna, Italy
b Università degli Studi di Bologna, Scuola di Informazione Via Sacchi 3, Casena, Italy

Received 10 August 1992
Revised 14 September 1992

Abstract


In this paper we present an expert system to make the real time diagnosis of an experimental apparatus. To overcome the drawbacks due to the time constraints required by a real time expert system a diagnostic apparatus based on a transputer network has been developed. The parallelization involves different parts of the diagnostic system: (1) the sensor data acquisition (2) the data interpretation, (3) the expert diagnosis. VME and PC boards have been used and the software has been written in Oznam, parallel C and parallel Prolog. Experimental results of the system performances are presented.

Keywords: Real-time diagnostic system; particle physics experiments; transputer networks; Oznam; parallel prolog; distributed database.

1. Introduction

Several real-time diagnostic systems based on a single processor have been developed to check experimental apparatus anomalies. The time constraints required by a real time expert system suggest to use a parallel architecture to perform a fast search of failures. We have considered an apparatus for particle physics experiments that, nowadays, are more and more complex and introduce remarkable problems for the on-line diagnostic [1]. To overcome the drawbacks a diagnostic system based on a transputers network has been developed. We have considered a particle detector composed of a set of gas filled tube streamer planes supplied by high voltage (HV). The signals from the wires are directed to different channels of electronic read-out modules, which are plugged in several acquisition crates. The stored information is available to a main acquisition computer.

2. Diagnostic system structure

The developed diagnostic system is structured in three levels:
(1) sensor data acquisition,
(2) data interpretation,
(3) expert diagnosis.

The complete structure is shown in Fig. 1.

Correspondence to: R. Campanini, Dipartimento di Fisica dell’Università degli Studi di Bologna and INFN Via Irnerio 46, 40126 Bologna, Italy.
At the data acquisition level the information coming from a high number of sensors is taken and handled by using suitable procedures and VME standard modules. A spy VME card performs monitoring and it stores in memory at each read-out cycle all the data flowing through the acquisition crates towards the main computer; this is done inserting this monitor module in each main acquisition crate.

A transputer in the commercial INMOS B011 VME card reads the data in the VME memory. This processor (VMET) has three links connected to the on-board C004 switch and a link to configure this switch. The C004-switch allows the connection between the VMET and a parallel network that performs the data interpretation level. This network is realized by commercial INMOS B008 card plugged in a IBM compatible personal computer and B014 card plugged in a VME rack.

3. Parallel data interpretation

The data interpretation network has been parallelized and it is composed of more transputer pipes. The first processor (Ring Processor = RP) of each pipe is connected by means of a C004 switch to the VMET and ring-connected as shown in Fig. 1.

Each part of the apparatus is related to a database of significant parameters useful for a statistical analysis. The data coming from the VME spy module contain the information to update continuously the database. The database dimension grows with the apparatus complexity and with the requirements of the statistical analysis. To optimize the memory
occupation and the load balancing of each computing node the database is distributed over all
the pipe processors (PP) of the network.

The RP processors collect the data according to the information contained in a subset of
the database. These processors send each group of data to the pipe which is able to elaborate
the data. These pipes perform the interpretation level extracting, with statistic elaboration on
the collected data, significant parameters useful for the diagnostic activity. To synchronize the
VMET activity with the data read-out, interrupt signals and counters are used. When a
process running on the VMET processor acknowledges an interrupt, it reads the data from
the dual port memory of the VME spy module. Moreover, another parallel process sends the
data, through the links, to the RP transputers with available memory space. The data coming
from the detector is decoded by the transputers in the pipes. The handled data is compared
with values representing the standard equipment status. Values which do not match the
standard conditions are sent to the expert system (ES), at fixed time intervals or at ES
request.

4. Parallel expert system

The data interpretation network is connected by means of another C004 switch to the
network implementing the expert system.

The expert system is written in a commercial multiprocessor version of CS-Prolog
distributed by Multilogic. This language is based on Hoare's Communicating Processes concept
[3], and it is a parallel language for use on environment with message passing architecture like
a transputers network. In CS-Prolog it is possible to assign a process to a normal Prolog goal
and then execute it in parallel with other processes, as a normal Prolog program. The
communication and synchronization of processes is made by messages exchange.

The parallel implementation of the expert system has been performed in two ways:
(a) by a processor farm model,
(b) by a network with specialized workers.
In both the configurations two kind of processes are always involved: master and worker. In
the model (a) the master performs the data acquisition from the data interpretation level
system and sends a fraction of these data to each worker. In case of (b) the data interpreta-
tion system broadcasts directly the list of suspected components to the worker subsystem. In
this case the knowledge base is distributed into the network. Each worker can infer a
diagnosis only about a specific class of elements. When all the partial diagnosis are done, they
are sent to the master which correlates and displays them.

Comparing the two models we found that in model (a) the master is overloaded and the
data input time is greater than in model (b). For these reasons we will give results only for
model (b).

The discussed experimental apparatus is composed of wire planes where each wire (W) in
the tubes is connected to the front end electronics (FE); the signals from the wires are
directed to different channels (CH) of electronic modules (RO), which are plugged in several
acquisition crates (CR). This acquisition structure can be represented with a tree, as indicated in
Fig. 2.

To perform the diagnosis on the apparatus the suspected wires are analysed [2]. For each
wire we have more elements to check, in agreement with the connections structure previously
described. All these elements are grouped in different classes according to their function.
Rules are made for each class and optionally can be written for each element of any class. An
element or class of elements is declared in 'failure state' if at least $N$ of the related wires are
suspected. For example, if we want to define the failure condition for the read-out module $X$,
we must write:

$$\text{Read Out}(X) \text{ is failed if at least } N \text{ wires are suspected.}$$
These rules may be deleted, added, and changed during run-time. It allows a high specialization of the system on a particular experimental setup.

5. Considerations about the parallel architectures

We have analysed the behaviour of the parallel data interpretation system (level 2) when it is composed of one pipe with \( np \) processors (Fig. 3). Each PP executes three different operations:

(i) routing of the data coming from the RP,
(ii) updating of the local database,
(iii) statistical analysis and routing of the results through the pipe to the ES.

Fig. 3. A data interpretation level with one pipe of length \( np \).
The time $T_i^i(np)$ spent from the $i$th processor to make these jobs can be described with:

$$T_i^i(np) = T_{\text{routing}}^i(np) + T_{\text{update}}^i(np) + T_{\text{stat}}^i(np) \quad i = 1, \ldots, np$$

(1)

where $T_{\text{routing}}^i(np)$, $T_{\text{update}}^i(np)$ and $T_{\text{stat}}^i(np)$ are the times to perform (i), (ii) and (iii) respectively. If the database is equally distributed over all the pipe processors only the fraction $M/np$ of $M$ input bytes from RP is, on average, associated to each PP, then

$$T_{\text{update}}^i(np) = (M/np)T_a \quad i = 1, \ldots, np$$

(2)

where $T_a$ is the mean time spent to update one byte. The $T_{\text{routing}}^i(np)$ time is proportional to the dimension of the input block data and to the fraction of this sent to the next processor:

$$T_{\text{routing}}^i(np) = M(1 - (i - 1)/np)T_i + M(1 - i/np)T_0 \quad i = 1, \ldots, np$$

(3)

where $T_i$ is the mean time to input, test and decide if a byte is localized on its own database or not; similarly $T_0$ is the mean time to output a byte. If we consider that the input and output process are practically in parallel on the same processor the time $T_{\text{routing}}^i(np)$ is given by the greatest of these times, in our case is:

$$T_{\text{routing}}^i(np) = M(1 - (i - 1)/np)T_i \quad i = 1, \ldots, np$$

(4)

To evaluate the time $T(np)$ spent by the $np$ processors in a pipeline to analyse $M$ bytes coming from the VMET the slowest stage of the pipeline has been considered. It is easy to demonstrate that in our case the slowest processor is the PP$^1$, because:

$$T_{\text{routing}}^1(np) > T_{\text{routing}}^2(np) > \cdots > T_{\text{routing}}^{np}(np)$$

then $T_{\text{routing}}(np) = T_{\text{routing}}^1(np) = MT_i$.

Moreover $T_{\text{update}}(np) = T_{\text{update}}^i(np)$ and $T_{\text{stat}}(np) = T_{\text{stat}}^i(np)$.

Then we obtain:

$$T(np) = T_{\text{routing}}(np) + T_{\text{update}}(np) + T_{\text{stat}}(np) = M(T_i + T_a/np) + T_{\text{stat}}(np).$$

(5)

We have also analysed the behaviour of the parallel ES (level 3). The high modularity of the structure suggest us a partitioning so that each worker contains the rules regarding a kind of element (specialized worker) (Fig. 4(a)) We have the FE workers, the CH workers, RO workers, and so on. Most of the work to make a diagnosis lies in the search in a list of rules. The mean number of steps for this search is:

$$((N/p) + 1)/2$$

(6)

in a chain with $p$ workers, where $N$ is the number of the rules.

When there is a fault in a system with this architecture, an FE worker has to scan a list with more elements than the CH worker. To overcome this drawback we can create a structure of pipelines where each chain is specialized (Fig. 4(b)). In this configuration we observe that, to have equal-sized tasks distributed to all the processing units, the parallel ES architecture must be mapped to the apparatus structure.

6. Results

The parallel system performance has been compared to the sequential solution as a function of the transputers number and network topology. The levels 2 and 3 have been
analysed separately, because the connection between these levels enables each one to operate independently. This condition has been very useful during the development and test phases.

The statistical data analysis is based on a poissonian distribution of the counts, and is performed with time steps of 15 s. In this case $T_{\text{stat}}(np) \ll T_{\text{runtime}}(np)$ and

$$T(np) = M(T_1 + T_0/np).$$

(7)
A transputer-based parallel expert diagnostic system

![Graph showing data rate versus number of processors](image)

**Fig. 5.** Analyzed data rate versus pipe length.

Experimentally the number of bytes analyzed by the pipe in 1 second has been measured. This is the rate \( r(np) \) given by:

\[
 r(np) = \frac{M}{T(np)} = \frac{1}{(T_f + T_a)/np},
\]

where \( np \) is the number of the processors in the pipe. In **Fig. 5** we show the experimental data rates obtained with a database of 30 read-out channels, each of 384 wires. The theoretical curve is obtained with a best fit based on the least square method using Eq. (7). The obtained values for the parameters \( T_f \) and \( T_a \) are:

\[
 T_f = 1.1 \pm 0.2 \text{ ms} \\
 T_a = 4.5 \pm 0.8 \text{ ms}
\]

\[
 \chi^2/ND = 4.4/11.
\]

We observe that the theoretical model is a good approximation of the experimental data. With this model the curve is asymptotic to 900 Kb/s. An improvement can be achieved with more pipeline connected as shown in **Fig. 1**.

![Graph showing speedup versus number of workers](image)

**Fig. 6.** Speedup versus worker number.
To compare the parallel architecture performances of the ES (level 3) a sequential configuration based on a single worker has been developed. We evaluated, in each configuration, a mean time to reach the diagnosis. The speedup $Sp$ has been defined as:

$$Sp = T_S / T_p,$$

where $T_S$ is the time used by the single-worker program, and $T_p$ is the time spent to make the same diagnosis by a program running on $p + 1$ processors (1 master processor and $p$ worker processors). In Fig. 6 is shown the speedup to reach a diagnosis when 200 wires are out of the standard range.

The efficiency $E$, defined as:

$$E = Sp / p$$

(10)

is shown in Fig. 7.

The asymptotic behaviour of the speedup is due to the partition of the expert system work. Most of the work is a search in a list of rules. The mean number of steps for this search is $(N + 1)/2$ with 1 worker and $((N/p) + 1)/2$ with $p$ workers (Eq. 6.), where $N$ is the number of the rules. The theoretical speedup in this simplified model is

$$Sp = [p(N + 1)] / [N + p].$$

(11)

When $N$ is fixed the behaviour of $Sp(p)$ is asymptotic to $N + 1$.

We observed (Fig. 6) that the obtained speedup is very close to the theoretical value. This means that the parallelism strategy adopted is quite efficient for this kind of problems and that the communication time between the processors does not significantly affect the algorithm speedup.

References

