Nuclear Physics B (Proc. Suppl.) 16 (1990) 520
North-Holland

# A SEMI AUTOMATIC FORTRAN TRANSLATOR FOR THE FNAL SECOND GENERATION ACP

R.C. BALL AND B.P. ROE

University of Michigan - Ann Arbor, Michigan 48109 - U.S.A.

In preparation for building a 250 VUP second generation Fermilab ACP system at Michigan for the CERN L3 experiment, we have developed a semi-automatic pre-compiler to convert event oriented programs into parallel code for the ACP.

The Fermilab second generation ACP system hardware consists of a set of MIPs R3000 processors (15 VUPs/processor) in special boards in VME crates. Each processor runs a full UNIX system and the whole array needs no master. However, it is capable of communicating with a VAX VMS computer which can serve as a front end.

A set of FORTRAN callable routines has been developed by Fermilab to enable the user to use the parallel power of this array. When designing a new program these routines allow convenient and flexible communication between the various processors. However, for existing large programs it is a non-trivial effort to change them into a parallel form for ACP use.

We have helped automate this process by developing a pre-compiler which does at least a large fraction of the modification and in many cases almost all of the modification for event oriented programs. We divide a user program so that the bulk of the program is in one process which is replicated on the different processors. The binary event input to all processes is through a single separate process, as is the binary event output.

There are three parts to the pre-compiler action:

1. Give guidance to the program and add special features. This is done by the use of ACP command lines which are in the form of comment lines mainly in the main program.
2. Intercept all I/O statements and replace them with calls to appropriate I/O routines.
3. Add the general I/O routines, make up the Job Description File needed by the ACP, and add the generally needed ACP routines.

The ACP command lines tag various key points in the program and provide some options.

The command lines are used to mark the start and end points for initialization, summary, event read in, event write out, and data retrieval. In addition they supply some needed information on random numbers, numbers of events to run, and indicate when to increment the event number and when to add together the histograms from the various processes.

Event binary output and event binary input is sent to and from the many event processes by means of a single input or output process. The read/write statements are converted into do loops and the record is moved from/to buffers for I/O process communication.

A version of the Gheisha hadron monte-carlo program was tested. This is a program of approximately 30,000 lines. A total of only 10 ACP command lines were needed. At present we are simulating the ACP system on a MIPs M/120 station which contains a single R2000 processor.

For up to five class 2 processes running on the M/120 we find that the efficiency is about 97 % compared to the original Gheisha (but with the new random number routine). For eight class 2 processes there is an additional loss of efficiency of about 2 to 3 %, presumably due to memory swapping. A test using the much larger L3 simulation program, SIL3, based on GEANT is now under way. This research is supported by the U.S. National Science Foundation.