RESULTS OF PORTING REAL-TIME FRONT-END SOFTWARE TO LINUX

D. Anicic, T. Blumer, I. Jirousek, H. Lutz, A. Mezger
Paul Scherrer Institute, Villigen, Switzerland

Abstract

Increasingly less expensive, more reliable, state of the art, standard PC components, with the now stable operating system Linux, provide a solid base for a port of the Accelerator Control Front-End. The poster shows odds and evens of the project in all it's phases, from the analysis, through the implementation and testing, to the integration into the Accelerator Control operation. To better judge the results of the project, we provide the comparison against the existing Real-time Unix platform.

1 INTRODUCTION

Looking at the future needs for the accelerator control system one can not overlook the desire to achieve the better performance, which is easily managed by introducing faster CPUs. In order to keep the hardware costs of the whole system at an acceptable level, we would like to use inexpensive standard PCs, powered by an Unix like operating system, Linux.

2 OUR CONTROL SYSTEM

2.1 In general

In order to achieve good performance, we have implemented the distributed control system, client-server based, message oriented. The clients, mostly X-windows workstations used as operator consoles, run all the necessary applications. They communicate their request messages to the front-end computers (FEC), which perform the actual input-output. Load balancing is easily achieved in such a system by introducing additional client and/or server computers. For more details see [1].

2.2 Front End Computers

Front-end computers interface the client application requests to the underlining process, the accelerators, over the CAMAC field bus. All input/output is handled at the front-end level. We look at the front-end computers as a collection of services, as shown in Fig. 1, which perform similar, but logically different tasks. The services are:

- PIOser, for single value I/O
- BLKser, for block of values I/O (profiles, …)
- LOOPser, for CAMAC loop manipulation
- ILKser, for the security system (INTERLOCK)
- LAMser, a CAMAC Demand messages dispatcher
- CAMser, for direct CAMAC access (HW tests only)

The services accept the client requests, process them evoking (sometimes repetitive) series of field bus operations, the results of which are analyzed to produce the replies. The analysis may involve complex calculations, introducing quite big software overhead. In such a system we can benefit from faster CPUs.

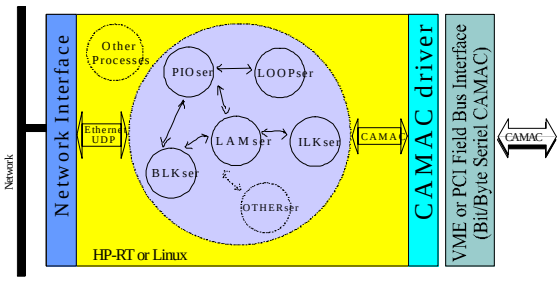

Fig.1: FEC as collection of Services

2.3 Services

The Services process client request messages. For historical reasons these are raw Ethernet messages. On the other hand the services within the same front end communicate between themselves (mostly LAMser with the others) using Unix message queue mechanism (inter-process communication, IPC). We also support the UDP/IP based communication. The implemented structure allows easy addition of any other communication mechanism. The Service also has to be able to process incoming requests and schedule them for further repetitive execution. The most natural approach was to implement Services as multi-threaded processes. Every receiving mechanism is handled by separate thread, requests are inserted in a common incoming queue which is emptied by the main processing thread. Requests for repetitive execution (yet another thread) are stored in a scheduled message queue. The queues are guarded against concurrent access by semaphores. In the initial design we had a strong emphasis to be portable, and therefore selected the POSIX standard, POSIX Threads and POSIX Real-time extensions as implementation strategy. Fig. 2 shows the Service's multithreaded structure.

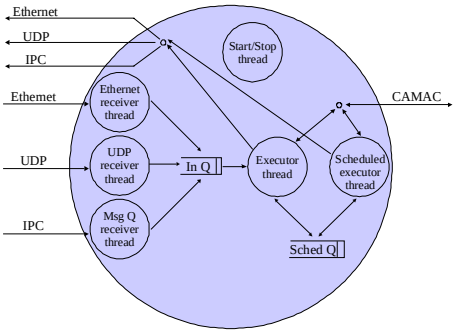

Fig. 2: Service multithreaded structure

3 CAMAC INTERFACE

3.1 Hardware

Front-end computers access the process equipment through CAMAC serial interfaces. At the time being our FECs are VME based HP rt743 running HP-RT (Real-time Unix). With them we use a CERN developed VME based CAMAC serial interface. For the PC-Linux port of the FEC software we tested two PCI based interfaces: the Kinetics System model 2115 (K2115) and Hytec Electronics model 5992 (H5992). We have found H5992 more useful for our needs even though the tested version was not complete, and are looking forward to test the final production version. The K2115 seems to be more suitable for data acquisition than control. It supports only execution of list of CAMAC commands, which is not optimally suited for our single command oriented approach while it introduces unnecessary overhead.

3.2 Software

For both, the present VME based rt743 FEC and PC-Linux port there is no CAMAC interface device driver available. Therefore we had to implement our own drivers.

4 PORTING

4.1 Preparations

As first step in porting our software to the Linux based PC we had to install the Linux operating system. It is a trivial task if one does not have to deal with all the newest PC hardware components (Ethernet, graphics, …). It was not our first Linux installation, so we had only some problems forcing Ethernet card in 100 MBit full duplex mode, but solved it by changing the Ethernet card. Afterwards it went easily, NFS mounted disks gave us direct access to the source code, and with the pre-installed GNU C compiler we were immediately in business.

4.2 Posix threads

Although Posix threads library should be the same on all the platforms, we did not found it such. There seem to be small discrepancies between the HP-RT and the Linux implementation. The minor differences are mostly in parameter passing mechanism (by reference vs. by value) and in very few cases different function names. The porting was done in less than two days, while we implemented the threads related code as macros defined in a single C header file.

4.3 Communication

Raw Ethernet access is very operating system specific, and in that sense not portable. The Linux kernel that we use (2.0.36) supports it only in promiscuous mode, so we did not port it. At the time of writing this article we have installed the newest Linux version (kernel 2.2.x) and found support for raw Ethernet. After few tests we estimate the port would not take more than a day. UDP/IP and IPC mechanisms required just slight modifications. The lack of raw Ethernet access did not cause us any problems. On Service startup configuration file is read and required or supported receiver threads are started. In this case we just had to disable the Ethernet receiver thread.

4.4 CAMAC driver

Drivers are not portable. They always have to be redone. But, since the present system is Real-time Unix and we were porting to Unix like Linux, the changes were not enormous. One of the differences is that present HP-RT system supports kernel threads (and we used them) and Linux does not. That means that interrupt handling kernel thread would have to be implemented as interrupt service routine. Interrupt service routine was not done while the serial CAMAC interface we mostly tested, H5992, does not support serial CAMAC demands (interrupts) properly yet. The rest of the driver code was ported in about one week, with additional week needed to understand how to handle either of the tested CAMAC interfaces. We had some worries about PCI plug and play mechanisms, but it turned out to be no problem at all.

4.5 Miscellaneous

Among other pieces of code to be ported comes the process memory locking (rt743 is a diskless single board computer with the NFS mounted disk) to avoid any possible swapping and/or paging. The code is slightly different, but it is a trivial modification. We have spent a great amount of time (about 20 %) because the GNU C compiler is stricter than HP C compiler. A lot of castings and function prototypes had to be inserted in a few dozen C files.

5 RESULTS

We were very pleased with the rapidness of the porting which took less than three weeks. It also resulted in our code being even more portable (casting and function prototypes mentioned above). We are also satisfied with the performance of the PC-Linux FEC. As an average we can assume an up to three times performance improvement, visible on Fig. 3. Here we have to mention that some (15 %) of the performance win is due to the 100 MBit full duplex Ethernet on PC-Linux against the 10 MBit half duplex on HP rt743. Of course there are some disadvantages, too. We have found that scheduled sleep on Linux can not take less than twenty milliseconds where we would prefer ten. Therefore maximal hundred times per second repetitive execution has to be reduced to fifty. Furthermore, Linux threads can not be scheduled with the accuracy and priority of the Real-time Unix. The result are the lags in request-reply closed loop times. We have observed the lags of up to 100 milliseconds. They occur in less than one in a thousand, and the overall performance
improvement highly compensates that.

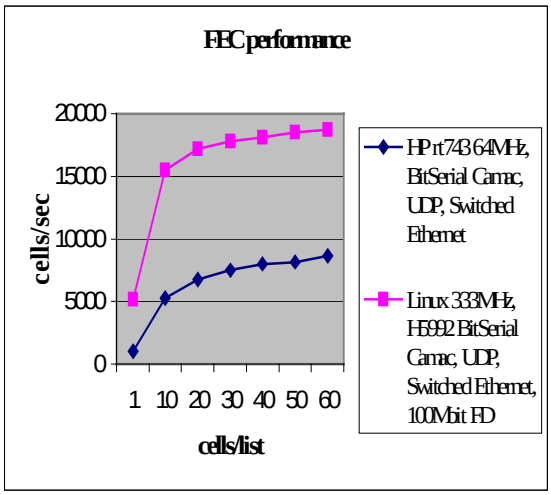

Fig. 3: FEC performance comparison

6 CONCLUSION

The so far good results should additionally be tested for long run stability of the ported software, the PC hardware and the Linux operating system in our control system as a whole. Altogether it seems to be a very good and an highly acceptable solution for getting additional processing power with less cost.

REFERENCES

[1] , T. Blumer et al, “Status Report of the PSI Accelerator Control System”, ICALEPCS’95, Chicago, Illinois.