SLAC-PUB--5929

DE93 004009

Real-Time on a "Standard" UNIX Workstation?*

Thomas Glanzman

Stanford Linear Accelerator Center, Stanford University, Stanford, California 94309 USA

ABSTRACT

This is a report of an ongoing R&D project which is investigating the use of standard UNIX workstations for the real-time data acquisition from a major new experimental initiative, the SLAC B Factory (PEP II). For this work an IBM RS/6000 workstation running the AIX operating system is used. Real-time extensions to the UNIX operating system are explored and performance measured. These extensions comprise a set of AIX-specific and POSIX-compliant system services. Benchmark comparisons are made with embedded processor technologies. Results are presented for a simple prototype on-line system for laboratory-testing of a new prototype drift chamber.

Introduction

A major new high energy physics initiative, a high-luminosity B Factory, has been proposed for construction at SLAC to study the nature of CP violation in the $B\overline{B}$ system.[1,2] This e^+e^- asymmetric collider will produce 10^9 events per year or an estimated 25 Terabytes of data.[3,4,5] It is hoped that funding will allow the machine and detector to be ready for collisions by late 1997.

Of the estimated 10,000 MIPS of processing power needed to analyze the data, approximately one-quarter is needed on-line for detector read-out, control, monitoring, logging and real-time event reconstruction. A decision toward "open systems" and the use of UNIX‡ everywhere in the experiment has suggested this challenging new use for general-purpose workstations. While part of the motivation for selecting UNIX as the only operating system was based upon financial and performance considerations, a strong argument can be made to minimize the number of different operating systems in the experiment. This can greatly improve the operability and reliability of the experiment by affording the contributing physicists a consistent computer platform for all phases of work. Basic operating system commands, the window environment, utilities, compilers and so forth are uniform and need only be learned once.

*Work supported by Department of Energy, contract DE-AC03-76SF00515

‡ Use of the word "UNIX" will, unless otherwise stated, refer to any of the UNIX-like operating systems provided by the major hardware vendors.

Presented at the 10th International Conference on Computing in High Energy Physics (CHEP), Annecy, France, September 21-25, 1992

The real-time needs of a B Factory experiment begin at the detector with digitization and trigger electronics and end once an event is logged to mass storage. In the current design a three-tiered trigger system is envisioned in which high-speed electronics and embedded processors are used for the first two trigger levels while the third level of trigger may reside in general purpose work stations. In addition, general purpose processors will be used for experimental control, monitoring, logging, etc. Level 3 processors must be able to cope with the estimated maximum 1 kHz interrupt rate, reading and writing of up to 25 kbytes of data and an as of yet unspecified amount of computation per event. Data logging processors must handle the design event rate of 100 Hz in which 25 kbyte events are read, processed (~25 MIPS-s/event) and written to mass storage. Sufficiently powerful workstations would allow the combining of the level 3 and logging processors. A goal of this R&D is to establish the limitations of currently available general purpose processors within the context of this scheme.

While the desirability of using powerful UNIX workstations for real-time applications may seem obvious, serious questions concerning their capability must first be answered. These questions span several broad areas:

- Real-time kernel and libraries (kernel preemptability, interrupt response, process control, interprocess communication, memory management, etc.)
- Distributed processing (network predictability/response, synchronization, interprocessor communication, etc.)
- External I/O (bus bandwidth and protocol flexibility, bus interfacing, etc.)
- Event builders (I/O bandwidth, buffer capacities, bookkeeping, debugging, etc.)

Each of these areas warrants a study of its own. This paper addresses only the first area. Two projects are being undertaken in this investigation. First, the basic attributes and tools needed by any real-time system are examined and performance measured. Second, these tools are applied to a simple prototype system to support an on-going drift chamber R&D project. As much additional work is necessary to implement a full-featured on-line system for a real experiment, this work represents only a start in this direction.

For these studies two IBM RISC System/6000 workstations were used, a model 520 and a 320H. The model 520 has a 20 MHz clock and is rated at 27.5 MIPS and 7.4 MFLOPS, while the 320H has a 25 MHz clock (since the architectures are similar, the performance numbers scale with the clock speed). Over the course of the work, several different versions of the AIX operating system were used, ranging from 3.1.3 through 3.2.2. Test results were not found to depend significantly upon the software version.

Real-Time and UNIX (AIX)

Overview

UNIX was not originally designed for real-time applications and its use in this area is due to the addition of significant extensions to the operating system kernel and libraries. Within the system libraries, one needs tools for handling external interrupts, process prioritizing,

interprocess communication, resource synchronization (e.g., semaphores), timer services, memory locking, memory sharing (both data and code), control over other system activity, etc. A predictable operating system kernel will, in addition, be fully preemptable. IBM offers these in its AIX version 3 operating system.[6] The supporting system libraries include functions coming from BSD and SVR3 UNIX, IBM-proprietary functions and (draft) functions from the POSIX 1003.4 standards committee.[7]

Open Systems

The developing family of POSIX standards promises to bring some degree of portability to real-time and other applications. IBM claims to be committed to these real-time standards once they are approved. In the meantime, AIX functionality reflects a very early version (draft 7) of this work. It may be that the 1003.4 balloting will begin soon and the standard approved near the beginning of 1993. Note that in addition to 1003.4, there are a number of other related standards such as 1003.4a (threads), 1003.4b and 1003.13. These are on a later time scale.

There may be compromises in the use of standard functions in that they must be executable on a wide variety of platforms, potentially limiting the user to the least common denominator of functionality. Consider the following AIX functions to set a timer "wake up" alarm:

Table 1:

Function	Library	Granularity	Comment
`tstart()`	AIX-specific	500 ns	10x clock interval
`incinterval()`	POSIX	10 ms	(draft 7)
`sleep()`	SVR3	1 s	
`select()`	BSD 4.3	1 s	

Note that the POSIX timer granularity (quoted here for nonprivileged users) does not reflect the machine's true capability. In 1003.4 draft 10, timer resolution is implementation defined and the programmer may request resolution information from the system. The availability of mechanisms to extract maximum performance through standard routines will encourage their use and should, therefore, be a high priority with the standards committee.

Scheduling and Real-Time Response

The AIX process scheduling mechanisms appear, at first glance, satisfactory for real-time applications. All runnable code may be divided into various categories. Interrupts are supported with 12 priority levels [0 to 11, 0 being most favored]. All normal processes effectively run at interrupt level 11. This means that all (enabled) interrupts run before any normal process. Normal processes run with either a fixed or adjustable priority. Typical interactive user processes run with adjustable priorities [40 to 127, 40 being most favored], while real-time and some system processes run with fixed priorities [0 to 39]. Note that the entire range of fixed priorities is more favored than the range of adjustable priorities. This scheme attempts to effectively separate the real-time and non-real-time processes. Additional details may be found in the IBM literature[6].

However, a normally functioning AIX system has its own lengthy agenda of tasks, bookkeeping and other miscellaneous activities which can easily interfere with real-time processes through the use of interrupts. For example, a normal system with an ethernet LAN connection is interrupted for each and every packet received. Such an interrupt may result in a high-priority daemon being scheduled. The daemon process may or may not interfere with real-time activity, but the interrupt processing is almost guaranteed to do so. Other common sources of interrupts include keyboard and mouse input, the virtual memory manager (page faults) and disk activity. These types of activity may be reduced (but not completely eliminated) by "quiescing" the system, that is, by killing all user processes, turning off network services, error logging and other functions. Of course, the system becomes single-user and completely isolated in this situation.

Another potentially troublesome source of interference originates with the system clock interrupts. AIX depends upon a steady "heartbeat" of system clock interrupts every 10 ms. At this frequency, an internal list is consulted and various tasks are executed. Tasks need not execute at every 10 ms interrupt. For example, competing interactive processes with the same priority may be switched at each 10 ms interval, while other scheduler activities occur every one second. By tracing all system activity over a period of several seconds, one can observe a number of different overlapping periodic activities with many different frequencies and CPU demands. Although individual tasks typically take only a small amount of time, say 100-300 ms, they reduce the predictability of the system's real-time response. Because this list of tasks is completely undocumented, there is the added uncertainty that future releases of the operating system may significantly change the real-time performance.

Typical Results

Three basic real-time metrics of interest which have been measured are listed below for the IBM/AIX workstation and compared with results from a recent SSC study[8] using a number of modern embedded processors with specialized real-time operating systems Note that the interrupt "service" latency refers to time between the CPU interrupt and interrupt handler, while "task" latency refers to the interval between the interrupt and first line of user code. Also note that the variation in numbers represents several effects. For the IBM machine the variations are due to different ways of measuring a given quantity plus variation within each measurement due to interference from system activity. The bottom row variations represent the summarizing of results from several different machines plus variation within each measurement..

Table 2:

Machine	Context Switch Time (μs)	Interrupt Service Latency (μs)	Interrupt Task Latency (ms)
IBM RS/6000-320H	40–263	57–690	270–4461
Various embedded μP + RTOS	5–50	6-88	100–343

Clearly, the performance of the IBM workstation is not on a par with the embedded processors. The IBM performance could be improved with a faster CPU (performance should scale with the clock), optimized kernel code and additional user control over system activity. As discussed earlier, control of system activity is crucial and may be the most important lesson from these numbers.

In the context of the B Factory level 3 trigger application, the IBM hardware used for these studies is approximately a factor of ten or more too slow from being a viable solution. However, industry anticipates significant hardware performance increases over the next few years, a typical claim being a factor of two every 12–18 months. IBM already offers a 50 MHz machine. Thus, machines available within three years may prove adequate for this task.

Prototype Test Bench

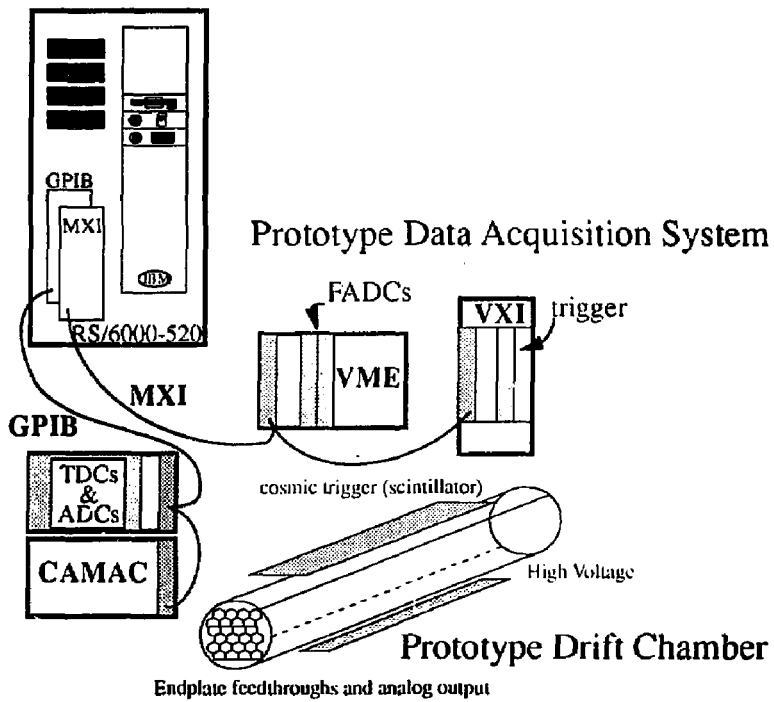

Figure 1. Prototype Hardware Configuration

A prototype system has been designed and implemented using the above tools to provide data read-out, control, logging and analysis for a 90-wire prototype drift chamber currently under construction at SLAC. The hardware components are illustrated in Figure 1. Two National Instruments interfaces connect the workstation with GPIB and VME/VXI crates. The goals of the prototype include: construct a data acquisition system meeting the needs of the prototype drift chamber; provide a prototype UNIX-based system for use by other B Factory detector development efforts; gain experience with VME and VXI buses; exercise a significant number of real-time tools available under AIX; test tape and disk I/O for data logging; and explore the tools and limitations of distributing this application across multiple processors. A schematic view of this hardware system is given in Figure 1. Data from each wire is digitized with LeCroy 2228–series TDC and 2249–series gated ADC electronics in one of two CAMAC crates. Up to

eight wires can be digitized in Struck DL515 250–MHz VME–based FADCs to measure ionization. The externally defined trigger is fed into a custom VXI module (built upon an Interface Technologies register-based prototype board) which passes the trigger on to the host computer as an external interrupt.

The software consists of five processes on two processors: read-out, control, logging, analysis and playback. In addition, control/data information is transmitted between processors via standard TCP/IP sockets and shared within the same processor via a shared memory segment. A system of semaphores is employed to control access to the data structures in memory. Logging can be done to disk or 8-mm tape. The initial software prototype system is currently in the final stages of development and is expected to be operational soon.

Summary, Conclusions and Future Plans

Some level of real-time response is possible using IBM's RS/6000 workstations running their real-time version of UNIX, AIX. The real-time tools and functionality are in place along side those supporting the normal multi-user mode of operation. Design of a real-time system must give careful attention to the degree of responsiveness required by the application. Real-time performance under AIX depends upon a number of issues: raw CPU power; system interrupt activity; and system process activity. Growth in raw CPU power is necessary for B Factory application but this seems a likely development in the time scale of interest. Control of system background activity is, at the moment, only partially possible - even in a completely quiesced system. The desirability of using a general-purpose workstation in a real-time environment is significantly diminished if it must be completely "quiesced" (single-user, no network services, X windows, etc.). These are the primary problems which will be pursued further and, hopefully, solved.

Other UNIX vendors have shown an interest in and are working on implementing true real-time capability on their machines. In combination with the POSIX standards, this "openness" should significantly expand the choice of compatible hardware for system design and future growth.

Real-time studies with the RS/6000 will continue in order to resolve these outstanding issues and to provide prototype code. It is planned to expand to the range of projects to include the other major areas of real-time interest. The prototype data acquisition system will continue to evolve: rewrite in C++ to investigate the use of OOP technology in the real-time environment; and add a graphical user interface.

Acknowledgments

The author would like to acknowledge the excellent contributions of two students, David Engberg from Stanford University and Jonathan Wong from San Francisco State University.

References

1. ``Investigation of an Asymmetric B Factory in the PEP Tunnel,'' SLAC–359 (March 1990),`` An Asymmetric B Factory Based on PEP: Conceptual Design Report,'' LBL PUB–5303/SLAC–372/CALT–68–1715/UCRL–ID–106426/UC–HRPA–91–01 (February 1991).

2. ``The Physics Program of a High-Luminosity Asymmetric B Factory at SLAC,'' SLAC–353 (October 1989).

3. ``Workshop on Physics and Detector Issues for a High Luminosity Asymmetric B Factory at SLAC,'' SLAC–373/LBL–30097/CALT-68-1697, 575-596 (March 1991).

4. T. Glanzman, ``Novel Aspects of the SLAC B Factory Computing Model,'' Proc. Int. Conf. on Computing in High Energy Physics '91 (Universal Academy Press, Inc., Tokyo, 1991) pp. 191–196.

5. D. Aston et al., ``Computing for a B Factory,'' SLAC BaBar Note 82 (June 1992).

6. IBM AIX Version 3.1, ``RISC System/6000 as a Real-Time System,'' Document number GG24–3633–0, IBM International Technical Support Center, Austin, Texas (March 1991).

7. ``DRAFT 10: Realtime Extension for Portable Operating Systems,'' IEEE committee P1003.4, sponsored by the Technical Committee on Operating Systems, IEEE, New York, February 1991.

8. K. Low et al., ``Overview of Real–Time Kernels at the Superconducting Super Collider Laboratory,'' Conference Record of the 1991 IEEE Particle Accelerator Conf., 91CH3038–7, v. 2, pp. 1308–1310.