KEK Preprint 95-84
July 1995
H/D

UNIDAQ, Real-time Response of the System

Y. YASU, M. NOMACHI, Y. NAGASAKA, R. BALL,
Y. TAJIMA and C. TIMMERMANS

5w9608

*Submitted to the RT95 Conference Issue of the IEEE Transactions on Nuclear Science,
East Lansing, MI, U.S.A., May 23 - 26, 1995.*

National Laboratory for High Energy Physics, 1995

KEK Reports are available from:

Technical Information & Library
National Laboratory for High Energy Physics
1-1 Oho, Tsukuba-shi
Ibaraki-ken, 305
JAPAN

Phone: 0298-64-5136
Telex: 3652-534 (Domestic)
(0)3652-534 (International)
Fax: 0298-64-4604
Cable: KEK OHO
E-mail: Library@kekvox.kek.jp (Internet Address)

UNIDAQ, Real-time Response of the System

Y.Yasu, M.Nomachi, Y.Nagasaka
KEK, Oho 1-1, Tsukuba 305, Japan

R.Ball
U.Michigan, The Harrison M.Randall Laboratory of Physics,
Ann Arbor, Michigan 48109-1120

Y.Tajima
Tokyo Institute of Technology, Meguro, Tokyo 152, Japan

C.Timmermans
148 Tate, Lab. of Physics, U.Minnesota, Minneapolis, MN 55455

Abstract

This paper describes a real-time response of UNIDAQ[1,2]. Interrupt task response time and task switching (context switching) time are measured for real-time UNIX (LynxOS) systems and standard UNIX systems. Here we show the correlation between the context switching time and the values of a benchmark program, SPECint92. We also show distributions of the response time for various operating systems (OS). A performance evaluation tool, DAQBENCH[3], was used for the measurements.

I. INTRODUCTION

UNIDAQ is a DAQ system based on standard UNIX and real-time UNIX. Standard UNIX is not a real-time OS, but its average response time with a RISC/UNIX system is fast[4,5,6]. Short average response time is essential for most DAQ systems. Therefore, standard UNIX can be used for the UNIDAQ. Observing the difference of DAQ performance between a standard UNIX and a real-time UNIX, we have investigated how the distribution of the interrupt response time is and how the context switching performs in comparison with SPECint92.

II. CONTEXT SWITCH VS. SPECINT92

The process communication in DAQ system based on multiple processes causes many context switchings. So, the context switching is an important DAQ function. DAQBENCH is used for the measurement of the context switching time. The programs execute semaphore system calls with context switchings and that without context switching. The context switching time is calculated from the time difference between the execution time of the two programs. The measurements were done on LynxOS systems (HP743, PowerPC601, i486DX2) and standard

UNIX systems (DECstation5125, SPARC2, AlphaXP3400). Figure 1 shows the number of context switchings per second in relation with SPECint92. The horizontal axis is the value of SPECint92 and the vertical axis is the number of context switchings per second. The context switching time on all of standard UNIX systems are proportional to SPECint92 values. That of all the LynxOS systems (in shadowed area) are also proportional to SPECint92 values. However, the slope of the LynxOS systems is 2 to 3 times steeper than that of the standard UNIX systems. LynxOS system gives better context switching performance than a standard UNIX system which has the same SPECint92 value. Hence SPECint92 is not appropriate in evaluating the DAQ performances.

Figure 1. Context Switch vs. SPECint92

III. INTERRUPT TASK RESPONSE

The interrupt task response time was measured on HP743/ LynxOS and standard UNIX (DECstation5125, SPARC2CE and AlphaXP3400). Figure 2 shows the setup for measuring the interrupt task response time. The time was measured by counting 1 MHz clock with a CAMAC scaler. The program generated a pulse first. This pulse was delayed for 100 msec through a gate generator and then the program went to sleep.

Measurement Procedure :

- Repeat those steps.
- (1) Clear Scaler Module
- (2) Write data on Output Register 1 to make pulse for Interrupt Register (The pulse is delayed by a Gate Generator for 100 msec.) (After the 100msec, Interrupt occurs and Scaler starts counting.)
- (3) Wait for an Interrupt from Interrupt Register
- (4) Write data on Output Register 2 to stop Scaler count (This process will wake up after the interrupt occurs)
- (5) Read Scaler for Interrupt task response time.

Figure 2. Setup for measuring Interrupt Task Response Time

```

CPU CONSUMER PROCESS ( INTEGER BASE )
main() {
    int i=0;
    while(1) {
        i++;
    }
}

CPU CONSUMER PROCESS( FLOATING POINT BASE )
#include <math.h>
#define N 1234567 /* odd number */
main() {
    int i=0;
    double a=0.0;
    while(1) {
        a += ((i%2)*2-1)*sin((double)((i++)%N)/N);
    }
}

I/O CONSUMER PROCESS
#!/bin/csh -f
while(1)
    ls -lR / >&/dev/null
end

```

Figure 4. Details of WORKLOAD

The delayed pulse was fed into a CAMAC interrupt register which generated an interrupt to wake up the program. The delayed pulse also made the scaler gate open and then the scaler started counting. After the program woke, it generated a pulse to close the scaler gate. Then, the program read out the scaler. The scaler values show interrupt task response time in μsec . Ten thousand events were acquired on each measurement.

Figure 3(a), 3(b) and 3(c) show the distribution of the interrupt task response time on the AlphaAXP with and without other workload. The horizontal axis is the response time in μsec . The vertical axis is the number of the events. As the workloads, two CPU consumer processes and an I/O consumer process were provided.

Figure 4 summaries the detail of the workloads. One of the three processes as a workload ran on each measurement.

Figure 3. Distribution of Interrupt Task Response Time

Figure 5. Distributions of Interrupt Task Response Time on LynxOS

In CPU consumers based on floating point operation or integer operation, the distributions were the same. The shape of distribution with CPU consumers in fig. 3(c) was almost the same as that without a workload in fig. 3(a). However, when the I/O consumer process ran, the distribution had a very long tail (see fig. 3(b)). The interrupt task response time over 100 msec was measured.

Result showed that the CPU consumer processes did not affect the response time, but I/O consumer process did affect the response time because the I/O consumer process executed many system calls and caused many interrupts.

Figure 3(b) and 3(d) show the distributions of the interrupt task response time measured in user-level and that in kernel-level, respectively. There is no difference on both distributions. Therefore, the interrupt task response time could be evaluated by the measurement in user-level.

Figure 5 shows the interrupt task response time on HP743/LynxOS with and without an I/O consumer process. The tail of the distribution with an I/O consumer process in fig. 5(b) is longer than that without a workload in fig. 5(a). One thing to note is that the maximum response time was extremely short when the priority of the measurement process (fig. 2) became higher than other workload processes.

The average time for the interrupt response on MVME167/ VxWorks was compared with that on HP743/LynxOS using the benchmark program in the DAQBENCH. The time on VxWorks was 131 μ sec, while that on LynxOS was 70 μ sec. The LynxOS system performed better than the VxWorks system.

The distribution of the interrupt task response time normalized for SPECint92 is evaluated below. The SPECint92 value of DECstation5125 was used as the base unit of the normalization. The histograms in fig. 6(a) are the distributions of the interrupt task response time with an I/O consumer process and the histograms in fig. 6(b) are

the same distributions as in fig. 6(a) except that its abscissa was multiplied by the SPECint92 ratio.

There were similarities between the distributions on standard UNIX, which had very long tails. However, LynxOS had a short tail and, as mentioned before, the tail disappeared when the process measured interrupt task response time gets higher priority than other processes. Therefore LynxOS had better DAQ performance of the interrupt task response time and guaranteed a maximum specified interrupt response time to DAQ system.

IV. CONCLUSION

The context switching time of standard UNIX systems and LynxOS systems were measured and evaluated using a SPECint92. SPECint92 was found to be inadequate for evaluating a DAQ performance. The DAQ benchmark, DAQBENCH, was good for the evaluation. LynxOS gave better performance of context switching than a standard UNIX which had the same SPECint92 value.

The distributions of the interrupt task response time on standard UNIX systems and on a LynxOS system, were measured. On standard UNIX systems, all distributions of the interrupt task response time had a similar trend. The distributions with an I/O consumer process had a very long tail. On LynxOS system, the distributions had a short tail, and its tail could be shorten when the process had a higher priority.

VI. ACKNOWLEDGMENT

We would like to thank Prof. Kondo, Prof. Watase and Prof. Takamatsu for their encouragement. We appreciate Prof. Fujii, Mr. Kodama and Mr. Inoue of the KEK On-line group for their help. We are grateful to Prof. Kato and his group members and Dr. Bazan for providing their computers.

Raw Data

Figure 6(a). Distributions of Interrupt Task Response Time with an I/O consumer process on standard UNIX and LynxOS

Raw Data * SPECint92 Ratio

Figure 6(b). Distributions of Interrupt Task Response Time with an I/O consumer process on standard UNIX and LynxOS

V. LIST OF TESTED COMPUTERS

HP742:	HP742/HP-RT(LynxOS) V1.11 (VME board computer)
HP743:	HP743/HP-RT(LynxOS) V2.01 (VME board computer)
PowerPC601:	CETIA(66MHz)/UNI-RT5.4 (VME board computer)
i486DX2:	DECPC(66MHz)/LynxOS 2.2
DECstation5125:	DECstation5125/Ultrix V4.3 VME I/F(T6000)
AlphaAXP:	DEC3400/OSF1 V1.3 VME I/F(T6000)
SPARC2:	SPARC2/SunOS 4.1.2 SPARC2/Solaris 5.3
SPARC2CE:	FORCE 2CE/SunOS4.1.1 (VME board computer)

VII. REFERENCES

- [1] M.Nomachi et al., UNIDAQ, Proceedings of the International Conference on Computing in High Energy Physics '94, LBL-35822(1994), pp.114-116.
- [2] UNIDAQ collaboration, UNIDAQ Document Set, SDC-93-573, Superconducting Super Collider Laboratory, 1993; UM-HE-93-29
- [3] Y.Yasu and Y.Tajima, Performance Evaluation Tool of DAQ Computer DAQBENCH, Proceedings of the International Data Acquisition Conference on Event Building & Event Data Readout in Medium & High-Energy Physics Experiments, Fermilab, Batavia, Illinois, U.S.A., October 26-28, 1994.
- [4] Y.Yasu et al., CAMAC Device Driver for DECstation with TURBOchannel VME adaptor - Is UNIX Real-Time Data Acquisition system possible?-(in Japanese), KEK Internal 91-7, December 1991
- [5] Y.Takeuchi, T.Tanimori and Y.Yasu, Development of Data Acquisition System using RISC/UNIX (TM) Workstation, Nucl. Instrum. Meth. A328(1993)526.; SDC-92-397
- [6] Y.Yasu et al., VMEbus based computer and Real-time UNIX as infrastructure of DAQ, Proceedings of the International Conference on Computing in High Energy Physics '94, LBL-35822(1994), pp.437-439.