Operating System studies for DAQ applications

G. Ambrosini[a], G. Mornacchi[b], C. Ottavi[b], L. Tremblet[b]

Abstract

The prototyping work being done for the data acquisition (DAQ) system of the ATLAS experiment at the LHC has provided the environment to explore the applicability of commercial operating systems to DAQ applications. LynxOS, a real-time UNIX-like operating system, and the WindowsNT operating systems have been studied from the point of view of the application environment by using an existing DAQ system.We report results and conclusions on the above studies which have been performed on VME processor cards.

1. Introduction

In the context of the ATLAS Data Acquisition (DAQ) prototype -1 [1] the need to select processing hardware and an operating system (O/S) has arisen. To this end we have performed an evaluation exercise whose objective was to study the combined performance of a hardware platform and the operating system in the context of Data Acquisition applications.
This paper summarizes the results of two independent evaluation studies ([2], [3]): on one side the LynxOS [4] operating system performance and on the other side the WindowsNT performance (the latter are only partially available). We emphasise that no attempt has been made to make a comparison between LynxOS and WindowsNT.
Although the studies have been performed on a broad variety of VME based hardware platforms, here we focus on the two most interesting cases: the PowerPC 604 (clocked at 100 MHz) based RTPC 8067 [5] running the LynxOS 2.3 operating system and the Pentium based VMIVME-7587 [6] running the WindowsNT operating system. The same benchmarks used on the PowerPC 604 running LynxOS have also been applied to a MIPS 3000 based VME board [7] running the LynxOS derived operating system EP/LX [8]. This provides a comparison with the DAQ environment currently exploited in test beam DAQ systems by ATLAS.
We have based our performance evaluation procedure on two kinds of benchmarks: single parameter benchmarks (where single features are measured in an independent way under different load conditions) and the performance study of a complete DAQ system (which, because in an application context, provides the most interesting performance information). To this end we have ported the RD13 DAQ system [9] to the PowerPC platform running LynxOS and the SPIDER DAQ system [10] to WindowsNT. In this paper we report on a sample of the measurements performed during the studies, more complete information can be found in [2] and [3].

2. Single Parameter Benchmarks

2.1 CPU Performance

Simple benchmarks such as the Dhrystone and Whetstone programs [11] provide an estimate of the raw performance of the processors used during the evaluation. We show in Table 1 the dhrystone performance results when the program is compiled with the optimisation option. We note that there is close to a factor of 4 in processing power

Table 1: Dhrystone performance (instructions/s)

	RTPC-604	VMIVME 7587	RAID-8239
Dhrystones/sec	208117	165000	56242

between the 604 and the MIPS processors.

a. Laboratory for High Energy Physics, University of Bern, Bern, Switzerland.
b. CERN, Geneva, Switzerland.

2.2 Semaphore Measurements

Semaphore performance has been measured using two basic micro-benchmarks: a single process lock/unlock cycle on a semaphore and the synchronisation of two processes (ping/pong test) via two semaphores. The latter, combined with the single process lock/unlock result, also provides an order of magnitude estimate of the context switch time. Measurements for both the "standard" System V semaphores and the O/S specific real-time ones have been performed.
Different CPU load conditions (i.e. no load, a CPU bound program running at equal priority and the same CPU bound program running at lower priority) have been used while measuring the performance of the single process lock/unlock benchmark.
Table 2 shows the results for the single process lock/unlock (in the case of the O/S specific real-time semaphores) expressed in microseconds. It should be noted that the values shown represent the total time corresponding to a lock plus an unlock call (i.e. two semaphore operation calls). The behaviour is as expected by a scheduler which is fair in time sharing mode ("same priority" (SP) column) and respects priorities ("lower priority" (LP) column).
We remark that the PowerPC results do not scale as much as could be expected when comparing the PowerPC and MIPS CPU power (see Table 1).

Table 2: Single Process Lock/Unlock (elapsed time in µs)

Platform	0 Load	Load at SP	Load at LP
RTPC-604	9.5	18.9	9.5
VMIVME-7587	14	N/A	N/A
RAID-8239	19.6	39	19.3

2.3 Context Switch Time

The context switch time was estimated as 1/2 of the difference between the 0 load ping/pong time and twice the 0 load lock/unlock time. The context switch time results are reported in Table 3.
When referring to the bare CPU performance (Table 1), we remark that the context switch time scales poorly for the PowerPC based system when compared to the less powerful MIPS 3000 system.

Table 3: Context Switch Measurements (elapsed time in µs)

Test Name	RTPC-604	VMIVME-7587	RAID-8239
Context Switch	8	13.5	13

2.4 Interrupt Handling Measurements

VME generated interrupts handled by the standard operating system software (the device driver for the LynxOS case and the IOWorks package [12] for WindowsNT) have been used for the purpose of studying the interrupt related behaviour.
We define *interrupt latency* as the time between the generation of the signal producing the VME interrupt and the execution of the first instruction in the user Interrupt Service Routine (ISR) and *task dispatch time* as the time between the raising of the kernel semaphore in the user ISR and the completion of the blocking wait in the user program.
Due to the late availability of the VMIC card, in the case of WindowsNT we have only performed the measurement of the interrupt latency. For LynxOS we have undertaken a sophisticated study of the interrupt system behaviour. For example we have obtained distributions of interrupt latencies and task dispatch times by using external timing logic. Detailed results are available in [2], while here we only present those results which can be compared with the values obtained with WindowsNT.
The LynxOS VME driver allows the use of two mechanisms to wait on an interrupt: an *ioctl call* directly to the device driver (this allows waiting on a single interrupt per thread) and the use of the standard *UNIX select system call*, which allows waiting on several file descriptors (hence several interrupts) at the same time. The latter mechanism, while more flexible, is inherently more time consuming. We measured the interrupt latency and task dispatch time, with both mechanisms, on the RTPC board running LynxOS.

EP/LX allows a user program to attach directly an ISR to a VME interrupt, without the need to go through a device driver. Since multiple interrupts on EP/LX have to be dealt with in a user defined way, we only show the results related to the single interrupt per thread case.
The above results are summarised in Table 4 in terms of both average values and widths.
The values from Table 4 show that the RTPC/604 and the RAID 8239 have similar performance for handling interrupts, which is not what one would expect on the simple basis of the CPU performance scaling.

Table 4: Interrupt Latency(IL)-Task Dispatch(TD) (μs)

	Ioctl (IL)		Ioctl (TD)		Select (IL)		Select (TD)	
	Average	RMS	Average	RMS	Average	RMS	Average	RMS
RTPC-604	9	3	9	3	11	1	67	2
VMIVME 7587	10.5	N/A	N/A	N/A	N/A	N/A	N/A	N/A
RAID-8239	14	0.5	16	4	N/A	N/A	N/A	N/A

3. Data Acquisition Timing Studies

3.1 The RD13 DAQ system on LynxOS

To study the PowerPC/LynxOS combination we have used as a macro-benchmark the RD13 DAQ system, originally developed to run on the RAID 823x platforms with the EP/LX operating system. A description of the working principles of the RD13 DAQ kernel (also called data flow protocol), along with indications of which and where system services are used, can be found in [2]. We made both a detailed timing study of the data flow protocol inside the detector read-out module and global performance measurements in terms of data acquisition rates. The results have been compared with those obtained when running on the RAID8239 under EP/LX.
We have selected from the many distributions in [2] the one showing the interrupt latency experienced by the DAQ system when running with a continuous and high (1 MHz) trigger rate (figure 1).
During our study we have used different data acquisition configurations, in particular one consisting of two tasks, one is an interrupt driven read-out (without burst structure) and the second a dummy event recorder. Event rate results for such a configuration are shown in Table 5 for both high (i.e. saturating) and low trigger rates. As expected, the PowerPC/LynxOS system performs better when the trigger rate is high (several 10s of KHz), where the CPU power effect is dominant. At low trigger rates (a few KHz) the PowerPC and the MIPS systems perform very similarly, indeed (see [2] for details) at low trigger rates the effect of O/S operations dominates (we have already remarked in section 2 that single O/S operations, e.g. semaphore flips, do not scale with the CPU power).

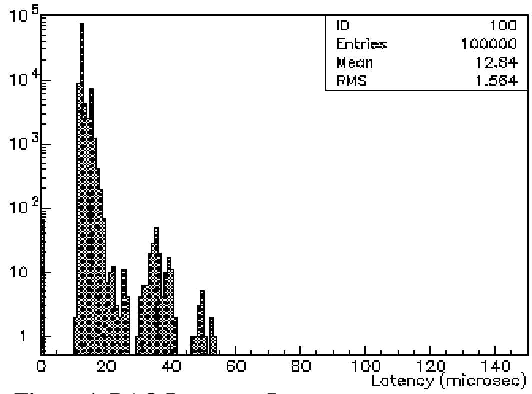

Figure 1. DAQ Interrupt Latency

Table 5: Total RD13 DAQ rates at high and low trigger rates

	RTPC (1 MHz rate)	RTPC (2 KHz rate)	RAID (1 MHz rate)	RAID (2 KHz rate)
Rate (events/sec)	~5500	~2000	~3500	~2000

3.2 The SPIDER DAQ system on WindowsNT

SPIDER is a simple and efficient DAQ package targeted to test beams and small experiments [10]. It was designed with hardware and software portability as a major concern. Indeed it has been ported to a variety of platforms and has also been used as a realistic tool to evaluate their performance.

SPIDER is entirely written in C and consists of about 15000 lines of C code (DAQ core and libraries included) and the SPIDER to Kernel Interface (SKIN). SKIN provides semaphores, shared memory access, process and priority scheduling, inter process communication and asynchronous notifications as well as miscellaneous services. This is the part which needs porting when moving to a new operating system.

When porting to WindowsNT existing UNIX dependent applications (e.g. when using the fork system call), the OpenNT subsystem [13] is potentially the most appropriate tool. The current lack of real-time and thread support in OpenNT, however, forced us to map the SKIN facilities directly on WIN32. The SKIN semaphore and shared memory concepts have been mapped onto WIN32 mutexes and memory mapped files. The SKIN process and priority scheduling required some work around the WIN32 CreateProcess call and usage of WIN32 real-time class priorities. The SKIN asynchronous notification was the most difficult mechanism to implement without having to modify the SPIDER source code: we have associated a thread and a semaphore per SKIN notification to take advantage of the multithreaded nature of WindowsNT[a].

The porting exercise completed, we have performed some performance measurements on the VMIVME-7587 board. When SPIDER is software triggered (i.e. without an external trigger such as a VMEbus interrupt), a maximum of 6600 events/sec can be processed. This result is good compared to the 5000 events/sec obtained with LynxOS 4.2 running on the same board. When triggered by real VMEbus interrupts, which were easy to handle owing to the powerful VMEmanager library [12], a maximum of 3300 events/sec can be processed when a user level ISR is used (interrupt latency of 170 us). An optimised version based on kernel-level ISRs (interrupt latency of 13.5 us) is in development and a number in the range of 6000 to 12000 events/seconds is expected.

4. Conclusions

The combination of a PowerPC processor and the LynxOS operating system has shown to be a good solution as the initial basis for our DAQ applications. Nevertheless we remark that LynxOS does not fully exploit the available processor power.

WindowsNT has shown to be a viable solution for DAQ applications which have no extreme real-time requirements (e.g. soft real-time applications).

The availability of WindowsNT subsystems such as OpenNT allows easy porting of existing UNIX applications and profiting from both the UNIX and the PC worlds.

5. References

[1] G. Ambrosini et al., The ATLAS DAQ and Event Filter Prototype “-1” Project, these proceedings.

[2] Atlas DAQ: Performance Evaluation of PowerPC VME Boards Running a Real-Time UNIX System, G.Ambrosini, G.Mornacchi,C.Ottavi. http://atddoc.cern.ch/Atlas/Notes/Welcome.html.

[3] Atlas DAQ: A role of WindowsNT and PCs in ATLAS FrontEnd DAQ, L.Tremblet. http://atddoc.cern.ch/Atlas/Notes/Welcome.html.

[4] DOC-0080: Lynx RTS, LynxOS 2.3.1 for PowerPC, Development Operating System.

[5] DOC/8067/UM: C.E.S. RTPC 8067 PowerPC Single Board Computer, Technical Manual.

[6] DOC-500-007587-000 B: VMIVME-7587 Pentium based VMEbus CPU Product Manual, VME Microsystems International Corporation.

[7] C.E.S. RAID 8239, 40 MHz VME RISC Processor Board, Technical Manual.

[8] CDC: Control Data Corporation, EP/LX User’s Guide.

[9] RD13-TN: The RD13 DAQ System (User’s Guide), S.Buono et al.

[10] CERN/ECP: The SPIDER user’s Guide Version 1.0, Y. Perrin.

[11] Dhrystone 2.1, SIGPLAN Notices 23, 8 (Aug. 1988), R. Weicker.

[12] DOC-520-009420-910 B: IOWorks Access User’s Guide for the WindowsNT Operating System.

[13] OpenNT for WindowsNT, Softway international, http://www.softway.com/OpenNT.

---

a. A drawback of this technique is the potential proliferation of threads.