Performance Comparison of VxWorks, Linux, RTAI, and Xenomai in a Hard Real-Time Application

A. Barbalace, A. Luchetta, G. Manduchi, M. Moro, A. Soppelsa, and C. Taliercio

***Abstract*—We report on a set of performance measurements executed on VMEbus MVME5500 boards equipped with MPC7455 PowerPC processor, running four different operating systems: Wind River VxWorks, Linux, RTAI, and Xenomai. Some components of RTAI and Xenomai have been ported to the target architecture. Interrupt latency, rescheduling and inter-process communication times are compared in the framework of a sample real-time application.**

Performance measurements on Gigabit Ethernet network communication have also been carried out on the target boards. To this purpose, we have considered the Linux IP stack and RTnet, an open-source hard real-time network protocol stack for Xenomai and RTAI, which was ported to the considered architecture.

Performance measurements show that the tested open-source software is suitable for hard real-time applications.

***Index Terms*—Real-time systems, Linux, RTAI, Xenomai, ADEOS, RTnet, VxWorks, PowerPC.**

I. Introduction

REAL-TIME feedback control has been extensively used in the RFX-mod experiment since 1998 [1]. RFX-mod is a magnetic-confinement, toroidal device, located in Padova, Italy, for studies on thermonuclear fusion. Seven control units are currently in operation and each unit consists of a VME crate hosting a Motorola MVME5500 single-board computer along with other boards for ADC/DAC conversion and digital I/O. The control units form a control network, where data are exchanged in real time among units. A software framework [2] has been developed to provide common functionalities for data handling and communication using the VxWorks platform [3]. VxWorks is widely used in physics research for several reasons, among which:

- It provides an integrated development platform, thus simplifying the development process. Programs can be developed and simulated in the host system before downloading them to the target system.
- It provides a powerful multitasking environment. Tasks have a fixed priority and can communicate via a rich set of inter-process communication (IPC) mechanisms.
- The software model of VxWorks is quite similar to that of UNIX, in particular for I/O and networking, thus simplifying software writing for developers who have experience with UNIX.

In recent years, however, the growing penetration of Linux has generated much interest in the possibility of adapting Linux for real-time control. Whereas for data acquisition and, more in general, for all those tasks that do not require strict timing determinism, Linux is already the standard in scientific experiments, there are some aspects of Linux that prevent a straightforward usage in real-time applications, among which:

- *Dynamic priorities*. Using a priority for processes that varies over time is for sure a good solution in time sharing, but may prevent an urgent process to get CPU ownership as soon as it requires it.
- *Paging*. It may introduce unexpected delays unless the page is locked in memory.
- *MMU remapping*. Remapping the Page Tables entries in the Memory Management Unit (MMU) when a user process gets the CPU may prevent fast context switching.
- *Coarse-grained synchronization*. Due to the non-pre-emptive kernel, the system may not react to events for some time in the case of lengthy kernel operations.

The recent Linux kernel 2.6 provides solutions to some of the above problems. It is in fact possible to associate a fixed priority with a subset of processes, and the kernel has been made pre-emptive by accurately defining un-interruptible segments in the kernel code and protecting them with spin locks, rather than disabling interrupts. Moreover, a new O(1) implementation of the scheduler has been provided in Linux kernel 2.6. Considering also that swapping can be disabled and that, for given level ranges, priorities can be made fixed, we can state that Linux can currently be considered a soft real-time operating system and can therefore be used for many applications which tolerate occasional delays in system response. However Linux 2.6 is not yet suitable for hard real-time applications, as required in feedback control for fusion devices. In this case, in fact, unpredictable response time may deteriorate the quality of the control or, worse, lead to unrecoverable instabilities.

Nevertheless, Linux has been successfully used for hard real-time control in [4] and [5], thanks to the pulsed nature of the experiments. In this case, for most of the time, the process control system does not need to operate in real-time mode. Real-time response is required only during plasma discharges which in most current fusion experiments last for few seconds. Therefore it is possible to disable interrupts when real-time behaviour is required, thus achieving an overall jitter in cycle time of 1–2 μs. When interrupts are disabled, extra care must be taken: the code and data must be locked into memory to prevent page faults, and the software cannot rely on system services for I/O. Therefore the control code cannot communicate with the external environment except for polling for input data and writing output data,

Manuscript received May 7, 2007; revised July 5, 2007. This work was supported by the European Communities under the contract of Association Euratom/ENEA. Paper no. TNS-00166-2007.

A. Barbalace and M. Moro are with the Department of Information Engineering, University of Padova, Padova, Italy.

A. Luchetta, G. Manduchi, A. Soppelsa, and C. Taliercio are with the Consorzio RFX, Euratom-ENEA Association, 35127 Padova, Italy (e-mail: cesare.taliercio@igi.cnr.it).

Digital Object Identifier 10.1109/TNS.2007.905231

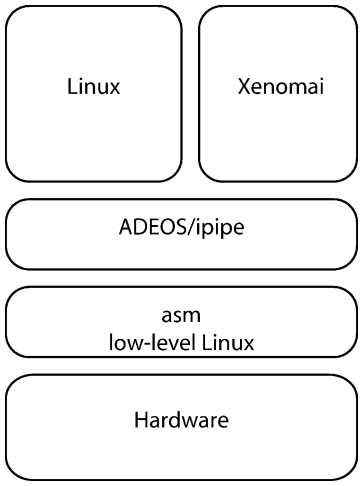


Fig. 1. Layering in Xenomai.

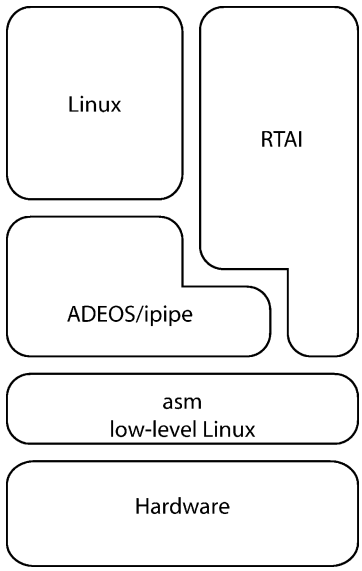


Fig. 2. Layering in RTAI.

using pre-configured address windows, for a fixed number of iterations.

This solution, although successful in particular applications, cannot be considered a general one for a variety of reasons, and most importantly because in the next-generation fusion devices the control system is expected to handle long-duration discharges, or even quasi steady-state processes.

II. Real-Time Extensions for Linux

What we are basically interested in is a way of adding to Linux the possibility of defining a few real-time tasks that are ensured to get control within a deterministic time as soon as they are ready to run. This feature is provided by two open- source Linux extensions: RTAI [6], [7] and Xenomai [8]. RTAI and Xenomai share most concepts (they originated from the same project) and both represent, rather than a replacement of Linux, an additional component that works in conjunction with Linux, handling the scheduling of real-time tasks and letting Linux provide all the remaining functionality.

In order to co-operate with Linux it is however necessary that the underlying hardware be shared by Linux and the additional component. This is achieved in both RTAI and Xenomai by using the ADEOS nanokernel [9], [10], which acts as a broker of the hardware functionality. In particular, hardware interrupts are normally handled by ADEOS, which propagates notifications in sequence to the other components. In this case Linux and RTAI (or Xenomai) represent ADEOS domains, and are logically organized by the nanokernel as a pipeline.

The component that is declared to be at the head of the pipe will receive interrupt notifications first and may then decide whether letting ADEOS propagate them along the domain pipe. In this case RTAI (or Xenomai) is at the head of the pipe and has therefore precedence over Linux, thus allowing deterministic response times regardless of the actual Linux implementation. This organization is fully achieved in Xenomai as illustrated in Fig. 1.

RTAI has a somewhat different organization, as shown in Fig. 2. Instead of letting ADEOS handle all the interrupt sources, RTAI intercepts them, using ADEOS to propagate those interrupt notifications to Linux in which RTAI is not interested in (i.e., the interrupt does not affect real-time scheduling). The reason for this mixed approach is performance, because in this case, if the interrupt is going to awake a real-time task, the overhead due to the management of the interrupt by ADEOS is avoided.

Both RTAI and Xenomai have an active developer community and both may represent an open-source alternative to VxWorks.

In order to assess the suitability for our purposes, we executed a set of measurements comparing the performances of Linux, RTAI, Xenomai and VxWorks. We used the PowerPC architecture, using in particular the Motorola MVME5500 single-board computer. The reason for this choice is that the PowerPC architecture proved to be better performing in floating-point computation than the x86 architecture. Moreover, in the development of the real-time control system of RFX-mod we had a very positive experience using the Vector Architecture Component (AltiVec) provided by the PowerPC processor family for parallel floating-point computation.

However, RTAI had not been ported to PowerPC and ADEOS had bugs which prevented its usage on our platform. Therefore we had first to develop a set of patches for using RTAI and Xenomai (both layered over ADEOS) on the MVME5500 board.

III. Porting Linux, ADEOS, RTAI, and Xenomai to the MVME5500 Platform

We have developed several patches for porting Linux, ADEOS, RTAI, and Xenomai to the target board.

For the Linux installation, a patch available from Motorola has been applied to a Vanilla Linux Kernel v 2.6.14, but we had to develop an additional patch in order to fix some bugs in VME data access. The nanokernel ADEOS had already been ported to the PowerPC architecture, but there was a bug preventing its usage for all platforms using the Galileo GT-64260 system controller, which has been fixed in a new patch developed by us. Xenomai has been installed with no changes because the version for PowerPC was already available and there was no need for specific board support, being Xenomai completely layered over ADEOS. No recent RTAI version was instead available for

PowerPC, and therefore a further development was required. Finally, the porting of RTnet did not require any additional development.

The patches developed for ADEOS and RTAI have already been integrated in the official distribution of both systems. All the patches for Linux on the MVME5500 board are available at the RFX-mod web site [11].

Since the target board is diskless, the boot of the system is first carried out by the factory-provided bootloader, Motload, which uses the Trivial File Transfer Protocol (TFTP) for downloading the kernel from a server machine. Network File System (NFS) is then used by Linux for file I/O.

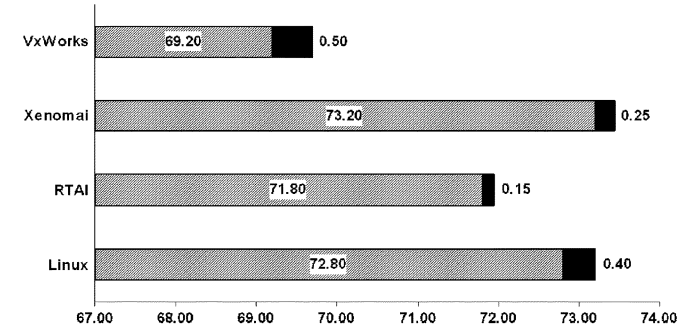
IV. Performance Measurement

For evaluating the performance, we concentrated on two features: interrupt latency and rescheduling time. We used a test program, *acqloop*, which acquires 64 cannels from an ADC converter via the VME bus and produces a single output on a DAC board, corresponding to the first acquired signal. Although very simple, this program allows testing the features we are interested in, since the ADC generates an interrupt when a new set of digitized data is ready. By measuring with an oscilloscope the delay between an input signal and the output of the system, we can perform a non-perturbative measurement of the overall system performance, thus allowing us to measure the differences in interrupt latency and rescheduling times. The other delays contributing to the overall delay are in fact due to the access time for VME I/O, which can be assumed to be the same for all the considered systems.

Finally, we considered real-time network communication since most control systems for fusion devices involve some sort of real-time communication. Both RTAI and Xenomai rely on the Linux IP stack for network communication. We provided therefore a first performance comparison between the IP stacks for VxWorks and Linux by extending our sample *acqloop* program to produce the output signal in another computer board, connected via Gigabit Ethernet and using the UDP protocol. We were interested in the UDP protocol because it is used for real-time communication in the current control system of RFX-mod. We then considered RTnet [12], [13], an open-source project for real-time network communication. RTnet provides a new implementation of the IP stack (up to UDP), where causes of non-determinism are accurately avoided. In particular, memory allocation for data packets is handled by pre-allocating all the required buffers in advance in order to avoid dynamic memory allocation. RTnet is available for both RTAI and Xenomai.

In a first round of tests, we focused on the differences in interrupt latency times among the considered systems. In this case, the program *acqloop* first installs an Interrupt Service Routine (ISR), which directly reads the ADC data register and writes to the DAC device (via VME). For VxWorks (v 5.5.1) *acqloop* is implemented as a C function which is then called by the shell, and all the code natively runs in kernel mode. For the other systems, this code is integrated into a Linux module, since it has to be executed in kernel mode. The overall measured delay is due to the times required:

1) by the ADC board to perform ADC conversion;



	Linux	RTAI	Xenomai	VxWorks
Jitter (µs)	0.40	0.15	0.25	0.50
Delay (µs)	72.80	71.80	73.20	69.20

Fig. 3. Measured delay and jitter without rescheduling.

2) by the VME interface chip in the computer board to detect the VME interrupt, and to propagate it to the system controller;
3) by the Operating System (OS) to handle the interrupt and to call the associate ISR;
4) by the ISR to read ADC data from VME;
5) by the ISR to write the output value to the DAC device;
6) by the DAC converter to output the corresponding voltage value.

Of the above times, only the third depends on the OS implementation. Therefore the differences in the measured delay highlight the differences in interrupt latency among the considered systems.

The measured times are reported in Fig. 3.

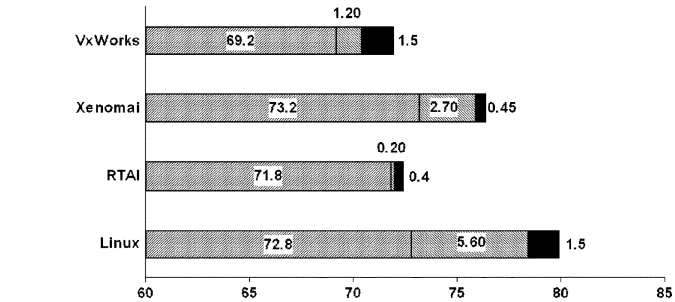
Overall the performance figures are similar with about 5% difference between the fastest (VxWorks) and the slowest (Xenomai). Closer analysis reveals that the difference between RTAI and Xenomai is a consequence of the fact that in Xenomai interrupts are always dispatched first to the nanokernel, thus introducing a small overhead, even with respect to Linux. Conversely, RTAI bypasses ADEOS in the management of interrupts, and relies on ADEOS only in the case the interrupt has to be propagated to another domain (Linux). It is worth noting that the performance of RTAI is very close to that of VxWorks and that, remarkably, the jitter appears to be smaller.

In a second set of tests, we changed the program *acqloop* so that the output is not written directly by the Interrupt Service Routine, but by another kernel task which is waiting for a semaphore, set by the ISR. In this case we take into account also the rescheduling task time. The results are listed in Fig. 4, where the second component of each bar represents the rescheduling time.

A first surprise has been the performance of Linux, both in delay and jitter. However such measures hold only for a system which is not loaded, and soon decrease when the workload increases. Conversely, the measured performances of the other systems proved to be only minimally affected by workload, unless involving a high interrupt rate.

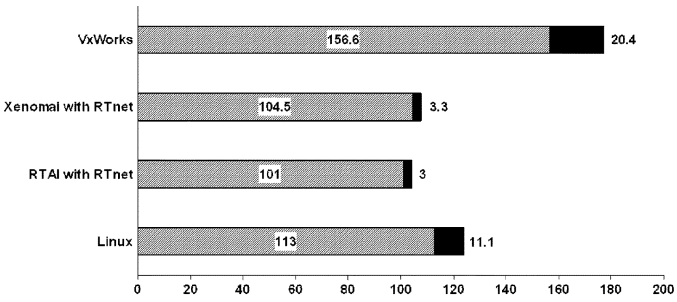
Again, the performances of VxWorks and of RTAI are very close. In particular it appears that RTAI has a very efficient scheduler for native real-time tasks.

In the last set of tests we consider network communication. In this case the process waiting for the semaphore does not write



	Linux	RTAI	Xenomai	VxWorks
Jitter (μs)	1.5	0.4	0.45	1.5
Rescheduling (μs)	5.60	0.20	2.70	1.20
Delay (μs)	72.8	71.8	73.2	69.2

Fig. 4. Measured delay and jitter including rescheduling.



	Linux	RTAI with RTnet	Xenomai with RTnet	VxWorks
Jitter (μs)	11.1	3	3.3	20.4
Delay (μs)	113	101	104.5	156.6

Fig. 5. Measured delays with real-time network communication.

the output directly to the DAC, but sends a UDP packet with all acquired data samples to another MVME5500 board, which in turn writes to the DAC converter. The size of the UDP packet is approximately 256 bytes (a few extra bytes are used for time-stamping UDP packets in order to detect packet loss).

The measured delays using network communication are listed in Fig. 5.

Here we compare the performance of Linux, using the native IP stack, RTAI and Xenomai, both using RTnet, and VxWorks, using its native IP stack. It is worth noting the poor performance of the VxWorks network communication, a fact we already experienced in RFX-mod using the latest version of VxWorks. RTnet shows very good performance with little jitter. Again, we observe a slight difference between RTAI and Xenomai, probably due to the different execution paths in interrupts.

In the above tests we enabled no access control disciplines for Ethernet in RTnet. We repeated the test with RTnet by enabling the Time Division Multiple Access (TDMA) access discipline on Ethernet. Such discipline is optionally provided by RTnet and aims at forcing determinism in network access, avoiding possible collisions. Using a TDMA cycle time of 100 μs and a time slot of 40 μs for the data packet, we obtained an overall latency of approx. 150 μs with a jitter around. 50 μs. It is not possible to reduce the TDMA cycle time (and therefore latency and jitter) since this causes an unacceptable CPU load.

V. Discussion

The aim of the reported work has been the evaluation of real-time Linux solutions for a possible replacement of VxWorks, currently in use at RFX-mod. Based on the reported performance measures, we observe the following facts:

- The performance of the current Linux 2.6 kernel is very good and may be acceptable in small, dedicated systems. This is however not the case for the feedback control system of RFX-mod, where the involved control units need to handle high data throughput in I/O and network communication.
- Both RTAI and Xenomai are worthy of consideration. Xenomai proved to be slightly less performing than RTAI, mainly because of its layered approach, which introduces some overhead in interrupt management. On the other hand, Xenomai is better structured and is available for a larger number of platforms. Moreover, Xenomai provides a set of emulation layers which may prove useful when porting large systems.
- Compared to VxWorks, both RTAI and Xenomai can be less user friendly for software developers. Since real-time tasks are to be executed in kernel mode in order to achieve best performance, the programmer cannot rely on the system services normally available in user space and debugging becomes very difficult. It is however possible, for both Xenomai and RTAI, to let user processes become real time. Allowing the development of user processes for real-time applications simplifies the development of real-time systems and permits also IPC with standard Linux processes. Real-time user processes are managed by a dedicated scheduler, which works in conjunction with the Linux scheduler by stealing user processes when they request to become real-time. Unlike kernel processes, context switching for user processes requires the remapping of the Page Table, a potentially time consuming operation. For this reason we plan further tests in order to quantify the impact of the MMU remapping in context switching.
- The network performance represents a strong point in favour of the migration towards RTAI or Xenomai. UDP has been successfully used for real-time network communication, and RTnet proved to be a very performing solution, especially compared with the poor performance we experienced with the latest version of the VxWorks IP stack for the considered board.

The best performance in RTnet is achieved without enabling the TDMA access discipline, which appears to be best suited to systems with a large number of access points (and therefore higher probability of access conflicts) but less stringent timing requirements. This is not the case of the RFX-mod experiment which involves less than 10 control units.

We can therefore state that both RTAI and Xenomai represent valid alternatives to VxWorks in the real-time control system of RFX-mod, and, more generally, for fusion devices.

References

[1] A. Luchetta and G. Manduchi, "General architecture, implementation and performance of the digital feedback control in RFX," *IEEE Trans. Nucl. Sci.*, vol. 47, pp. 186–191, 2000.

[2] M. Cavinato, G. Manduchi, A. Luchetta, and C. Taliercio, "General-purpose framework for real time control in nuclear fusion experiments," *Trans. Nucl. Sci.*, vol. 53, pp. 1002–1008, 2006.

[3] Wind River home page, [Online]. Available: http: www.windriver.com.

[4] J. A. Stillerman, M. Ferrara, T. W. Fredian, and S. M. Wolfe, "Digital real-time plasma control system for Alcator C-Mod," *Fus. Eng. Des.*, vol. 81, pp. 1905–1910, 2006.

[5] B. G. Penaflor, J. R. Ferron, M. L. Walker, D. A. Piglowski, and R. D. Johnson, "Real-time control of DIII-D plasma discharges using a Linux alpha computing cluster," *Fus. Eng. Des.*, vol. 56-57, pp. 739–742, 2001.
[6] RTAI Home page, [Online]. Available: http:// www.rtai.org.
[7] P. Cloutier, P. Mantegazza, S. Papacharalambous, I. Soanes, S. Hughes, and K. Yaghmour, in *DIAPM-RTAI position paper, Nov. 2000, RTSS 2000—Real Time Operating System Workshop*, 2000.
[8] Xenomai home page, [Online]. Available: http://www.Xenomai.org.
[9] ADEOS home page, [Online]. Available: http://www.adeos.org.
[10] Karim Yaghmour Opersys Inc., Adaptive Domain Environment for Operating Systems, 2001. [Online]. Available: http://www.opersys.com/ftp/pub/Adeos/adeos.ps.
[11] RFX-Mod home page, [Online]. Available: http:www.igi.cnr.it.
[12] RTnet home page, [Online]. Available: http://www.rtnet.org.
[13] J. Kiszka, B. Wagner, Dr. Y. Zhang, and Dr. ir. J. F. Broenink, "RTnet—A flexible hard real-time networking framework," in *Proc. 10th IEEE Int.Conf. Emerging Technologies and Factory Automation*, Catania, Italy, 2005.