Impact of the Linux Real-time Enhancements on the System Performances for Multi-core Intel Architectures

Nabil Litayem
Technologue Professor
ISET-Bizerete, Menzel
Abderahmène 7035

Slim Ben Saoud
Conference Professor
LECAP, EPT-INSAT, Centre urbain
nord, BP 676 Tunis cedex, Tunisia

ABSTRACT
Embedded Linux became a dominant choice in the embedded entertainment and mobile systems. Their adoption in widely used control applications is the second phase of their embedded market domination. One of the most important criteria of the control RTOS is their determinism/overhead ratio. Actually, many extensions exist to bring real-time capability into the Linux kernel. On the other hand standard computer architecture become widely adopted in the embedded market, with a large variety of performances and power requirement.

In this paper, we study the impact of timing enhancement offered by various real-time Linux kernel extensions and their impact into the overall system performance. The obtained results are compared with the standard and server kernels performances.

We used for our study a multi-core Intel® based architecture since we considered the trend of the embedded control market for this kind of architectures.

In our work we studied two metrics to reflect the performance of the studied kernel that are latency and throughput. Such work can be used to orient the adoption of real-time Linux extension for a given hardware architecture to reach control application requirements.

General Terms
Real-time, Benchmark, Linux

Keywords
Real-time Linux, Xenomai, LowLatency, PREEMPT-RT, control applications

1. INTRODUCTION
Modern control applications require much newer functionality like GUI (Graphic User Interface), communication possibilities and great software reusability. Traditional RTOS (Real Time Operating System) can reach the timing performance but have many weaknesses concerning nowadays required aspects. Various traditional RTOS tried to enhance their functionality by offering additional software components through additional costly licenses. On the other hand standard Linux kernel obeys the other's requirements but cannot be used as a hard real-time operating system.

These reasons impulses several initiatives to integrate real-time capabilities into the Linux kernel, which can make of Linux a very serious candidate in the embedded systems field. Actually, many approaches are available to offer these functionalities using different architectures [1], [2]. The most adopted solutions are RTLinux/RTCore, RTAI, Xenomai and PREEMPT-RT [2] patch. Each one of these real-time enhanced kernel has their internal architecture, their strength and weaknesses. The widely available choice in terms of timing performances and functionalities among different Linux kernel variants makes of Linux one of the most suitable embedded operating systems, widely adopted for different embedded applications with different constraints range.

Actually, PREEMPT-RT patch is finally mainlined in the current kernel and used by great real-time field actors such Wind River® in their Linux4 solution. Xenomai [3] is another successful real-time project widely adopted in hard real-time application. This extension is actually adopted by Sysgo company with their real-time Linux solution called ELinOS. Moreover, few works tries to merge Xenomai with PREEMPT-RT, in the original solution baptized Xenomai/Solo, which port Xenomai capabilities to PREEMPT-RT patch.

On the other hand, the embedded processing requirements are increasing at an exponential rate. The supply in terms of embedded processors is becoming increasingly broad. Different platforms can be adopted and used in the embedded field, classically FPGA and DSP architectures are widely adopted in the embedded high performance field. Actually, we assist in the convergence of PC and embedded architecture. Different conventional microprocessor actors try to enlarge their activities with processor which can be used in both standard and embedded computer. Intel® and AMD® with their respectively ATOM™ and GEODE™ processor are considered as interesting candidates in the embedded field. Other high end processors designed to desktop and server systems become adopted in industrial computer designed by many great embedded control actors such as Siemens® and National Instruments®. These processors are used by various manufacturers in industrial control or for hardware-in-the-loop applications. These kinds of architecture are often adapted to offer special robust peripheral enhanced to work in industrial environments.

In this paper, we try to investigate the usability of industrial computer for real-time control applications using various real-time Linux extensions. For this goal, we evaluate the real-time performance and throughput of Xenomai, PREEMPT-RT, Lowlatency, standard and server kernel. Timing performance are measured using Cyclictest, Unixbench are used for throughput evaluation. Our timing performance tests are released under a workload generated using hackbench benchmark. These

performance evaluation tools were adopted after a qualitative comparison of various tools.

This approach is adopted to study the timing performance of Linux and its impact on the overall system performance for both single and dual-core systems. The studied platform is a standard computer with CoreTM 2 Duo Intel® microprocessor, 3Go of DDR2 RAM and based on Ubuntu Linux 10.10. Such a platform is similar to high-end industrial computers designed for control purpose.

The following paper is organized as follows. Section 2 presents a survey of dominant real-time open source Linux solutions. Qualitative comparison of performance evaluation tools are presented in section 3. We studied the latency of different kernel versions using cyclictest time measurement program in Section 4. The system performance evaluation is presented in Section 5 for both single-core and dual-core system. Conclusions and discussion are related in Section 6

2. REAL-TIME LINUX EXTENSIONS

Computer system could be considered as a real-time system if the time is a dimension of the correctness. The most important aspect of such system is deadline meeting.

Linux is a general-purpose operating system designed for desktop and server usage. Its kernel was previously designed to guarantee the best resource allocation for all executed processes. Desktop and server Linux kernels use the CFS (Completely Fair Scheduler). This scheduler is not adapted to real-time systems since they are characterized by unfairness. Their successful adoption in these two fields pushed various embedded system actors to extend their usage into embedded systems field. Such adoption has a good recognized effect into the embedded field but the kernel hasn't the required timing performance for real-time applications. Many academics and industrials efforts were made and proposed to enhance the Linux kernel with real-time functionalities. Actually there are several existing implementations of real-time extension for Linux kernel [4].

2.1 Real-time Linux technology

Academic research and industrial efforts have created several real-time Linux implementations [5], [6]. These extensions can be categorized into two categories according to the approach used to improve the timing performance.

The first approach consists of modifying the kernel behavior to improve its real-time characteristics, by reducing the durations of high priority task.

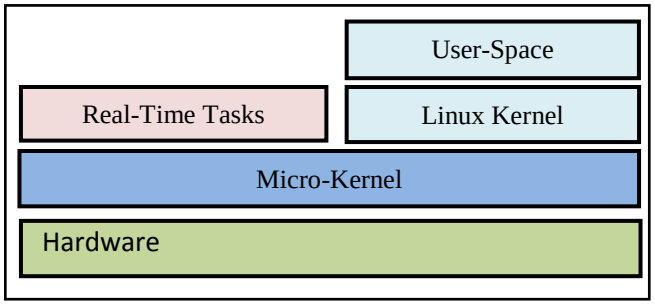

Fig. 1: Microkernel based real-time Linux

The second approach consists of using small real-time kernel to handle real-time tasks and who can run the Linux kernel as a low priority task. The idea behind this approach is illustrated by Figure 1. The most known projects using this technology are RTAI and Xenomai. These two projects are built behind ADEOS that allow the creation of multiple domains. ADEOS are also responsible for interrupt management, as every triggered interrupt is oriented to its registered domain. However, if one interrupts without knowledge of ADEOS is received by one domain it's systematically forwarded to the next domain in the ADEOS pipe. Figure 2 shows the interrupt management of ADEOS based real-time Linux.

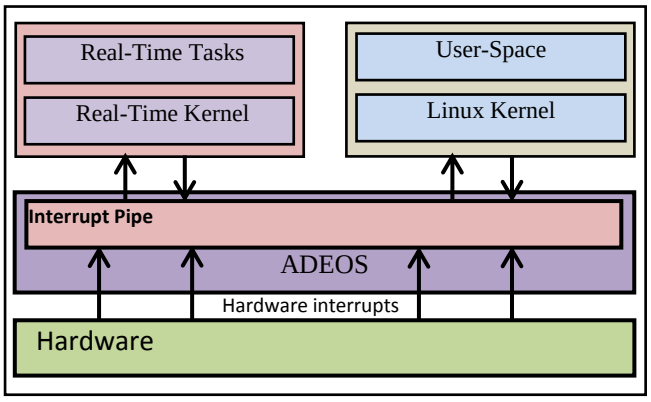

Fig. 2: ADEOS based real-time Linux

2.2 Main real-time Linux solutions

There has been noteworthy works to transmute Linux into hard or soft real-time operating system. These works are essentially based into one of the previously presented technology.

In this section the most popular implementation of these technologies will be discussed.

2.2.1 Preemptible Kernel (lowlatency)

This extension was previously developed as an external patch called preempt-kernel by Robert Love [7]. Since 2.5 kernel version preempt-kernel patch was incorporated into the mainline kernel to offer better reactivity qualities. Thanks to this extension every process may be scheduled out practically everywhere in the kernel.

This project was initiated by the transformation made to the Linux kernel for SMP (Symmetric Multi-Processor) support. Such support required the critical section protection from concurrent access to process running on distinct CPUs. This protection was realized using a spinlocks.

These spinlocks are used to protect the kernel areas from concurrent access. Such areas are nearly the same that must be protected to offer a reentrant kernel.

2.2.2 PREEMPT-RT

The PREEMPT-RT patch is the most successful Linux modification that transforms the Linux into a fully preemptible kernel without the help of microkernel [8]. It allows almost the whole kernel to be preempted, except for a few very small regions of code. This is done by replacing most kernel spinlocks with mutexs that support priority inheritance and are preemptive, as well as moving all interrupts to kernel threads.

(Dubbed interrupt threading), which by giving them their own context allows them to sleep among other things.

This patch presents new operating system enrichments to reduce both maximum and average response time of the Linux kernel. These enhancements are progressively added to the Linux kernel to offer real-time capabilities. The most important enhancements are:

- High resolution timers
- Complete kernel preemption
- Interrupts management as threads.
- Hard and soft IRQ as threads
- Priority inheritance mechanism

Some of these new features like Threaded IRQ are currently pushed to the mainline kernel by the patch maintainers.

2.2.3 RTAI

RTAI is a real-time application [9] interface usable for both uni-processors and symmetric multi-processors (SMPs). This extension allows the usage of Linux in many "hard real-time" applications. As an option, RTAI's "LXRT" allows the control of real-time tasks, using all of RTAI's hard real-time system calls, from within Linux memory-protected user space resulting in soft real-time combined with fine-grained task scheduling. RTAI is the real-time Linux that has the best integration with others open source tools scilab/scicos [10], [11]. This extension is widely used in control applications.

2.2.4 Xenomai

Xenomai [12] is a real-time development framework that can be integrated with the Linux kernel to provide hard real-time support. The current version is based on dual kernel approach. It implements ADEOS (I-Pipe) micro-kernel between the hardware and the Linux kernel. I-Pipe is responsible for executing real-time tasks and intercepts, interrupts, blocking them from reaching the Linux kernel to prevent the preemption of real-time tasks by Linux kernel. Figure 3 illustrate the functional behavior of the ADEOS/I-Pipe with the case of Xenomai implementation. The resulting system is composed from Linux and small co-kernel running side by side on the same hardware. Xenomai co-kernel exclusively controls the real-time applications and real-time interfaces either to kernel-space modules or to user-space applications.

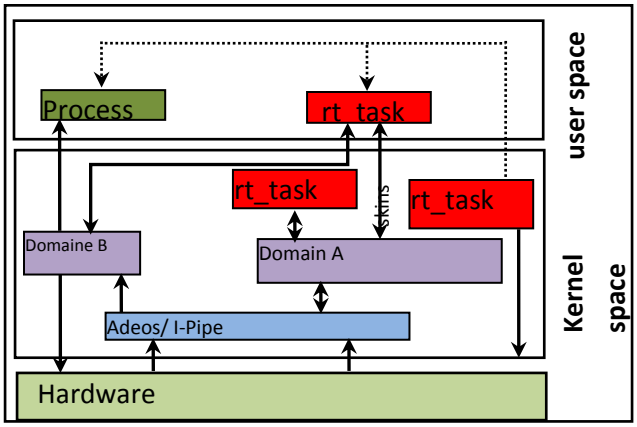

Fig. 3: Interrupt management in Xenomai

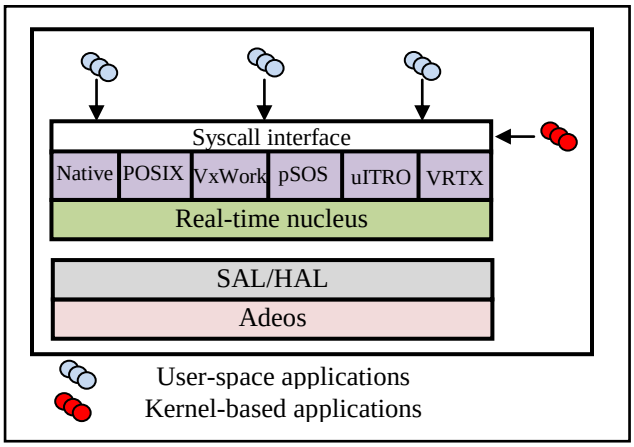

Fig. 4: Xenomai skins architectures

These interfaces called skins can mimic pSOS+, VRTX, VxWorks, POSIX, uITRON and RTAI API. Due to this feature, Xenomai was considered as the RTOS chameleon. It was designed to enable smooth migration from a traditional RTOS to Linux without having to rewrite the entire application. Figure 4 illustrate the Xenomai skins architecture and show that almost skis are equivalent to the Native skins.

On the other hand Xenomai support a wide range of architecture (PowerPC32 and PowerPC64, Blackfin, ARM, x86, x86_64, and ia64).

3. REAL-TIME PERFORMANCE MEASURING AND BENCHMARKING

Real-time computer system has three performance aspects that must be monitored to reveal the overall system performance [13], [14]. These three aspects are real-time performance, throughput and stability. Such work can be done using real-time measurement programs, benchmarks and stress tools. The obtained results are generally used to measure, analyze and improve both hardware and software architecture by manipulating various factors.

3.1 Real time measurement program

To reflect real-time operating system health various measurement programs exist. Each one has its approach and focalizes in a well determined performance aspect. Table 1, resume the most important real-time measurement programs. The most important feature for such systems is to provide determinism. Other features such response time, scheduler robustness, protection from priority inversion, offered preemption mechanisms etc., can be considered as quality metrics. Each one of these programs is able to evaluate a separate or a set of factors. However, the worst-case execution time and jitter can resume the overall timing performance. Cyclictest can be used to measure these two metrics by measuring the time between configured timer expiration, and the actual expire time. For this reason we decided to adopt Cyclictest for the rest of this work to reflect the real-time performance of our system.

Table 1. Real-time measurement program

Real-time measurement program	Description
Lpptest	Benchmark included in the PREEMPT-RT patch that measure the interrupt latency received on the parallel port.
RTMB	Micro-benchmark suite, designed to compare many of the common metrics of real-time performance across several platforms and several languages (C, C++ and Java).
RealFeel	ANSI/C program that test of how well a periodic interrupt is processed.
Cyclictest	ANSI/C program that measure the scheduling latency of the Linux kernel. Cyclictest recurrently goes to sleep for a certain time interval and measures the actual duration of the sleep to infer the latency.
LRTBF	A benchmarking Framework composed of a set of drivers and scripts for evaluating the performance of various real-time additions for the Linux kernel
Houglass	A synthetic real-time application that can be used to learn how CPU scheduling in a general-purpose operating system works at microsecond and millisecond granularities.
Senoner test	A latency benchmark designed to analyze the Linux behavior under under high system load.
Bytemark	CPU benchmark suite, reporting CPU, cache, memory, integer and floating-point performance

3.2 Benchmarking programs

Real-time computer system can be compared with their relative performance. This can be done by running a number of standard tests and trials against it. Benchmark program results are essentially dependents from the hardware but the software execution environment has a remarkable impact on the obtained results. The main purpose of Benchmarks is to offer a way of comparing the performance of several subsystems through different hardware/software architectures.

Each benchmark is able to cover various sets of system performances. In the context of Linux based computer system, various communities or industrial benchmark are available for different computing purpose. Table 2 resume the most used open source Linux Benchmarks.

Table 2. Open source Linux Benchmark

Benchmarking program	Description
hackbench	ANSI/C benchmark designed to measure the performance, overhead, and scalability of the Linux scheduler.
Lmbench	ANSI/C microbenchmarks designed to measure latency and bandwidth.
UnixBench	ANSI/C benchmark designed to provide a basic indicator of the performance of a Unix-like system. UnixBench can measure various aspects of the system's performance and support multi-CPU systems.
IOZone	ANSI/C filesystem benchmark that generates and measures a variety of file operations..

3.3 Stress programs

Stress programs are in general used to test the stability of a computer system in the building and tuning purposes. For the Linux kernel several stress programs are used to validate every kernel release. Each one of these programs, recapitulated in table 3 can cover several aspects of the kernel functionalities.

Table 3. Open source Linux stress program

Stress program	Description
dohell	Script based on previously presented hackbench benchmark and the dd command, that heavily load the entire system
Stress	Simple ANS/ C program that can impose a configurable amount of CPU, memory, I/O, and disk stress on POSIX-compliant operating systems
Calibrator	Small ANSI/C program designed to extract the cache memory, main memory and TLB parameters
Cpu Burn	Stress program, designed to heavily load CPU chips.

4. REAL-TIME CHARACTERISTICS

4.1 Timing performance evaluation

Measuring real-time performance of a Linux based operating system can require various aspect investigations of the studied system. The most important aspect of such system is WCET (Worst Case Execution Time) and Throughput. Table 1 show a recapitulation of the most adopted benchmarks and test programs.

Cyclictest benchmark can be used with different parameters to determine the latency of various samples or only the average and maximum latency. In our case, we used the verbose mode to study statistically the latency and the silent mode to determine the average and the maximum latency. The obtained results for the studied kernels are plotted in Figure 5 to Figure 9.

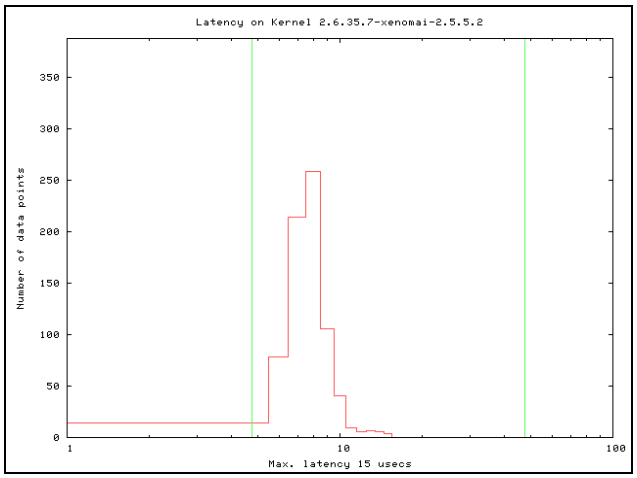

Fig. 5: Statistic latency results of the Xenomai patched Linux kernel

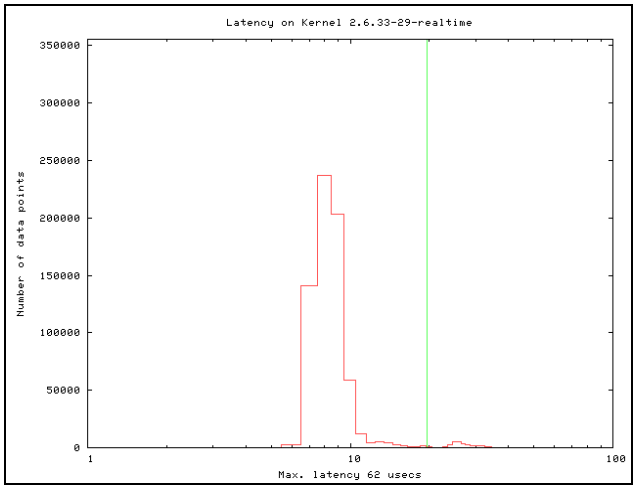

Fig. 6: Statistic latency results of the PREEMP-RT patched Linux kernel

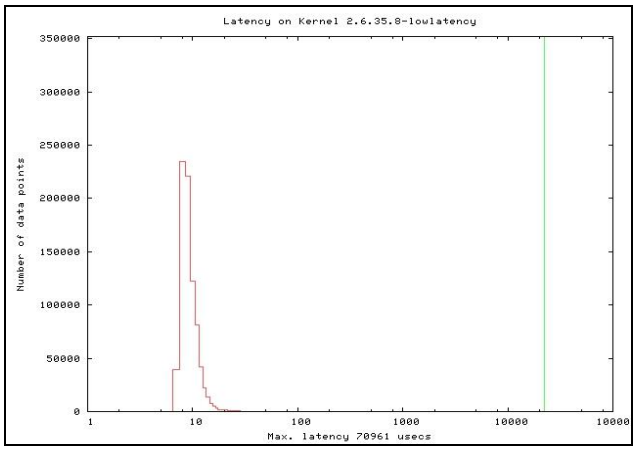

Fig. 7: Statistic latency results of the low latency Linux kernel

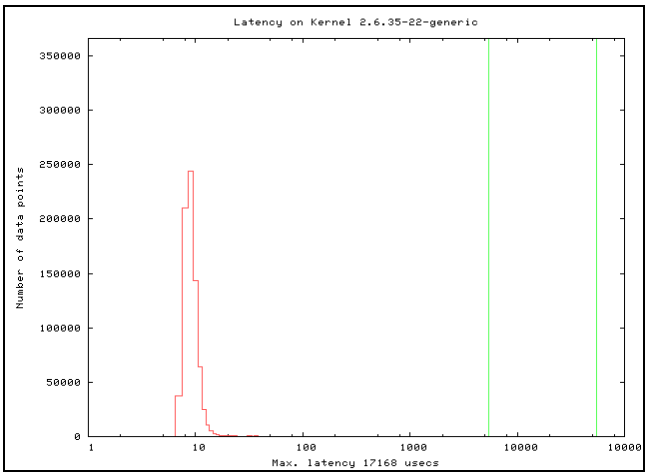

Fig. 8: Statistic latency results of the generic Linux kernel

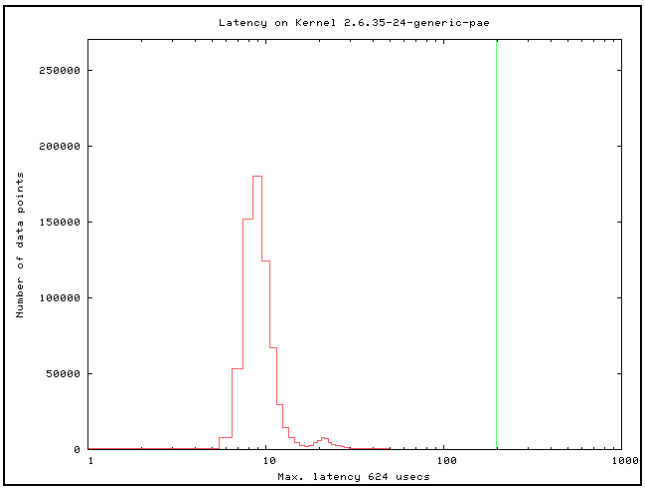

Fig. 9: Statistic latency results of the server Linux kernel

4.2 Interpretation

The earlier presented results show that the average response time of the five studied kernels is around 10µsec. The best maximum latency is obtained with Xenomai which are about 15µsec. This result can be justified by the architecture of Xenomai that separate real-time and Linux domains.

On the other hand the obtained result with PREEMPT-RT can encourage the usage of this kernel for hard real-time applications since the maximum latency is about 62µsec. The main advantage of such solution is their entire compatibility with the classic Linux applications.

Low latency kernel show better average results than the standard kernel but their maximum latency can be a serious limitation for its adoption in hard real-time applications.

Standard and server kernel are given as a reference result for other real-time enhanced kernel.

5. SYSTEM PERFORMANCE

5.1 System performance evaluation and benchmarking

New improvements in computer technology announce miscellaneous requirements and constraints for system performance evaluation, especially with the emergence of multi-core architectures.

System performance evaluation can be very helpful in the design-flot since it reflect the performance of a whole system, including all its aspects. Several system benchmark suites exist. This benchmark can be classified as follows:

- CPU benchmark
- Embedded and media benchmark.
- Language specific benchmark
- Transaction processing benchmark
- Web server benchmark
- Domain specific benchmark

Every benchmark from the presented categories should be representative of the applications that can run on the studied systems. These different categories and its relevant benchmark are detailed in the book [12].

Actually, with the convergence of desktop and embedded systems, system benchmark can be used. The most useful benchmarks in our case are LMBench, UnixBench and Nbench. We adopted UnixBench for the rest of our work since this benchmark is updated to support multiprocessor system and has a great portability under different UNIX® systems.

5.2 UnixBench

UnixBench is designed to extract a basic performance indicator of a UNIX® system. Various aspects of the system are reflected using an index to compare the performance of the current system to a reference system. The entire set of index values is then combined to make an overall index for the system. UnixBench can also handle Multi-CPU systems. The advantage of this benchmark in our study is its capability to reflect the performance of the overall system (including the operating system and used compiler) not only the available hardware which is the case of real systems.

The individual performance reports indicate the performance of the system in a different specific domain like integer or floating point computation. The system benchmark score indicates the performance of the global system. For the two reports, the upper score indicates better performance.

5.3 System performances using single processor and 2 cores

UnixBench can detect the various CPU available on the studied system and parallelize its different benchmark on these CPUs. The reported values are given for both single and multi-core configuration. Since we used a dual core processor they obtained results illustrated in Figure 11 and 12 are for single-core and dual-core.

The obtained results are an attributed score to the whole computer system, computed according the result of various internal benchmarks such as Dhrystone and Whetstone.

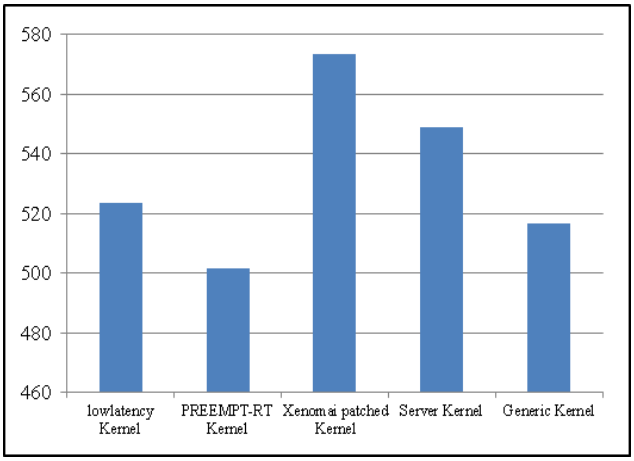

Fig. 10: System Benchmarks score for different studied kernel running under a single-core.

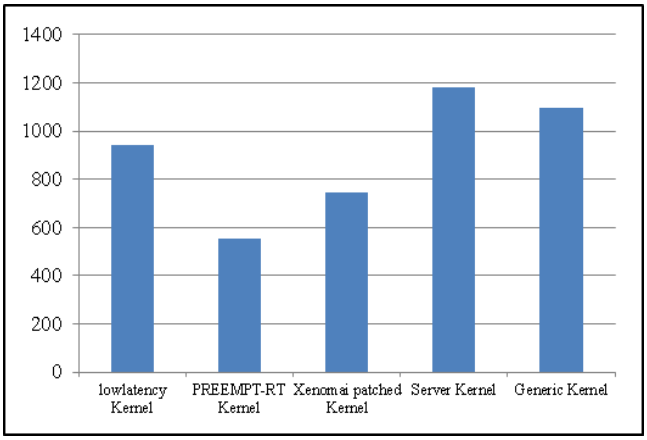

Fig. 11: System Benchmarks score for different studied kernel running under a dual-core.

5.4 Interpretations

The measured values for single-core architecture show global performances degradation caused by real-time capabilities for PREEMPT-RT and lowlatency kernel in the order of 16 % compared to the standard Linux kernel. Xenomai patched kernel is shown better global performance than standard Linux kernel. This result can be explained by the deactivation of the power management and frequency scaling in the Xenomai patched kernel. On the other hand the xenomai results are obtained with unloaded real-time domain.

The dual-core architecture shows a considerable degradation for the PREEMPT_RT patch in the order of 49% compared to the standard Linux kernel. More else, we can consider that the performance of dual-core architecture is 9% higher than single-core architecture for this kind of PREEMPT-RT kernel.

A wiser choice can be the adoption of higher performance single-core processor instead of dual-core. These kind of results can be explained by the maturity of PREEMPT-RT multi-core support.

6. CONCLUSIONS

This paper provided a comparative study of various real-time enhanced Linux kernels. Our results show that Xenomai and PREEMPT-RT have a comparable performance of mono-processor system with a little superiority of Xenomai booth for latency and throughput. We concluded that the adoption of PREEMPT-RT can be a wiser choice for monoprocessor real-time system due to the smooth migration of application development from standard Linux to PREEMPT-RT.

On the other hand the multiprocessor results show a clear degradation of the obtained results for PREEMPT-RT patch that can be an obstacle for the PREEMPT-RT adoption of such systems. Lowlatency Linux kernel can be a serious candidate for soft real-time application in a multiprocessor environment since this extension shows a good behave under such an environment, additionally LowLatency has the same programming model as standard Linux kernel.

As follows up to this work, we plan to investigate the capabity of Xenomai/Solo solution who try to merge PREEMPT-RT and Xenomai solution.

7. REFERENCES

[1] K. Yaghmour, G. Ben-Yossef, and P. Gerum, “Building Embedded Linux Systems”, O’Reilly 2008.

[2] M. Mossige, P. Sampath, and R. Rao,“Evaluation of Linux rt-preemptfor embedded industrial devices for Automation and Power technologies – Acase study”, In Proceedings of the Ninth Real-Time Linux Workshop in Linz, November 2007.

[3] P. Gerum, “Xenomai - implementing a rtos emulation framework on gnu/linux,” 2004, 2008.

[4] M. Tim Jones, “Anatomy of real-time Linux architectures From soft to hard real-time”, IBM 15 Apr 2008

[5] Z. Chen , X. Luo , Z. Zhang, “Research Reform on Embedded Linux’s Hard Real-time Capability in Application”, Embedded Software and Systems Symposia, 2008. ICESS Symposia '08. International Conference on 29-31 July 2008 Page(s):146 - 151.

[6] S. Level, “Anatomy of real-time Linux architectures From soft to hard real-time”, in IBM, 2008 ed: IBM, 2008.

[7] Arnd C. Heursch, Dirk Grambow, Dirk Roedel and Helmut Rzehak, “Time-critical tasks in Linux 2.6, concepts to increase the preemptablity of Linux kernel”, Linux Automation Konferenz, University of Hannover, Germany, March 2004.

[8] Dongwook Kang, Woojoong Lee, and Chanik Park, “Kernel Thread Scheduling in Real-Time Linux for Wearable Computers”, ETRI Journal, Volume 29, Number 3, June 2007.

[9] F. Jiang, S. Gao Jie Zhang, “A Hardware-in-the-loop Simulation System of Diesel”, Power and Energy Engineering Conference, 2009. APPEEC 2009. Asia-Pacific Engine Based on Linux RTAI 27-31 March 2009 Page(s):1 - 4.

[10] R. Bucher, S. Balemi,”Scilab/Scicos and Linux RTAI - a unified approach”, Control Applications, 2005. CCA 2005. Proceedings of 2005 IEEE Conference on 28-31 Aug. 2005 Page(s):1121 - 1126.

[11] G. Doukas, and K. Thramboulidis, “A Real-Time Linux Based Framework for Model-Driven Engineering in Control and Automation”, IEEE Industrial Electronics, Volume PP, 2009 Page(s):1-11.

[12] Byoung Wook Choi, Dong Gwan Shin, Jeong Ho Park, Soo Yeong Yi, Seet Gerald, “Real-time control architecture using Xenomai for intelligent service robots in USN environments”, Springer Intel Serv Robotics 2009 Page(s):139–151

[13] L. Kurian, J. Lieven Eeckhout, “Performance Evaluation and Benchmarking”, Published in 2006 by CRC Press.

[14] Paul J Fortier, Howard E. Michel , “Computer Systems Performance Evaluation and Prediction”, Digital Press 2003