Experimentación y Evaluación de Sistemas Operativos de Tiempo Real

Fernando Romero, Luciano Iglesias, Roberto Guisández,
Armando De Giusti, Fernando Tinetti[1]
III-LIDI, Facultad de Informática, UNLP
{fromero, degiusti, ftinetti}@lidi.info.unlp.edu.ar, li@info.unlp.edu.ar rguisandez@gmail.com

Abstract. El presente trabajo presenta experimentos con diferentes sistemas operativos que soportan las características de tiempo real. Dichos experimentos están destinados a medir y comparar el tiempo que dichos sistemas operativos demoran para responder a un evento externo con una respuesta. Este tiempo de demora se llama latencia. No solo se evalúa este tiempo, sino la variabilidad del mismo. Si bien no es la única condición que debe cumplir un sistema operativo para ser considerado de tiempo real, es una condición necesaria. Los tiempos de latencia y su variabilidad darán los límites de requerimientos de restricciones temporales que podrán soportar.

Keywords: Tiempo Real, Hardware, Sistemas Operativos, Métricas, Planificación de CPU, latencia.

1 Introducción

Los Sistemas Operativos de Tiempo Real (SOTR) [1] [4] [5] [6] [12] son aquellos que permiten proveer en forma garantizada un servicio dentro de un intervalo de tiempo de respuesta limitado. De acuerdo a los requerimientos de las aplicaciones de Tiempo Real, este intervalo puede ser desde el orden de los segundos, al orden de los microsegundos o nanosegundos. La corrección de un sistema de tiempo real, a diferencia de los sistemas convencionales, involucra tanto a las salidas que produce como el tiempo que demora en producirlas. Si no se satisfacen estas restricciones temporales en los tiempos de respuesta, se arriesga la falla completa del sistema.
La característica más importante en un SOTR es dar respuesta a través de un servicio ante eventos internos y externos, dentro de los plazos requeridos. Estos eventos incluyen interrupciones de hardware externas, señales de software internas e interrupciones de temporizadores internos.
Una medida para la capacidad de dar respuesta de un SOTR es la latencia, es decir, el tiempo que se demora entre la ocurrencia de un evento y la ejecución de la primera instrucción en el código del programa que da servicio a la interrupción. En el caso de una interrupción de hardware, la latencia es el tiempo que lleva desde que una interrupción de hardware requiere servicio hasta que comienza a ejecutarse la rutina

[1]Comisión de Investigaciones Científicas de la Provincia de Bs. As.

del servicio de interrupción. Esta latencia puede ser medida con instrumentación externa al sistema de cómputo, tal como osciloscopios, cronómetros, etc. Debe considerarse, al medir la latencia de interrupción, que la medida debería incluir cualquier demora introducida por el bus de periféricos del sistema. Tales demoras pueden deberse a competencia con otros periféricos que están transfiriendo datos sobre el bus y apropiándose del acceso al mismo.

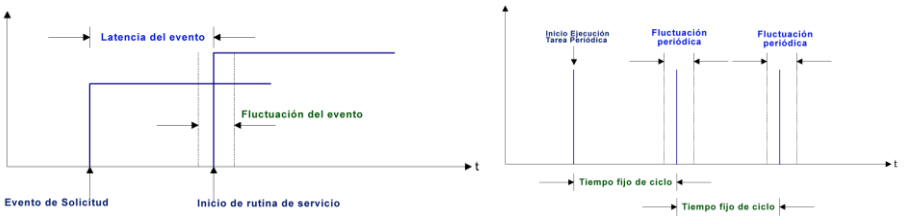

Fig.1: a) Latencia b) Fluctuación periódica

A los efectos de medir la aptitud de un SOTR [2] [18] para considerarse tal, se considera la latencia como el tiempo desde que la señal llega al SO hasta que la primera instrucción de la tarea es ejecutada. Esta latencia puede ser medida usando el registro del contador de reloj de la CPU (TSC, Time Stamp Counter). Este registro de 64 bits se incrementa en la frecuencia de reloj, y una secuencia de sus valores puede almacenarse en memoria para un posterior análisis. Esta técnica de medida no es afectada por ninguna demora de acceso al bus de dispositivos, porque el registro del contador es interno a la CPU y siempre se incrementa en cada ciclo de reloj. Una segunda medida es la fluctuación periódica (jitter) de dicha latencia, que es la variación en el tiempo de latencia que experimenta una tarea que se ejecuta reiteradamente. Esto es crítico para las tareas periódicas que se repiten ante una interrupción provocada por un reloj que mide el periodo.

2 Tiempo Real en Linux

Debido a que Linux es de código abierto, se ha utilizado para desarrollar SOTR [10] modificándolo a nivel de kernel. El objetivo de estas modificaciones es llevar la latencia de interrupciones y las fluctuaciones entre interrupciones periódicas al rango de los microsegundos, permitiendo así una rápida respuesta a eventos y una temporización de alta resolución. Las pruebas de este trabajo se han realizado sobre SOTR basados en Linux. Una de ellas, la metodología de apropiación del kernel consiste en modificar el kernel Linux estándar para asegurar que procesos de kernel con alta prioridad puedan apropiarse del uso de la CPU frente a otros procesos con prioridad menor. Esta modificación implica cambiar el manejador estándar para interrupciones de dispositivos de forma que las interrupciones de mayor prioridad no se vean bloqueadas por una cantidad de tiempo arbitraria mientras se maneja una interrupción de menor prioridad. A partir de la versión 2.5.4, el kernel estándar incorpora la lógica apropiativa. El planificador puede ejecutar una tarea de Tiempo Real con prioridad mayor apropiando cualquier tarea normal de menor prioridad en el espacio de kernel.

En Linux estándar se obtienen latencias en el orden de un milisegundo. Las versiones

de Tiempo Real de Linux tienen latencias y fluctuaciones en el orden de unos pocos microsegundos. Existen tres aproximaciones básicas para modificar el kernel estándar del sistema operativo Linux, para que provea respuestas de Tiempo Real:

•Micro-kernel: Se inserta una capa de código nueva, y altamente eficiente, entre el hardware y el kernel estándar, llamada micro kernel, que se hace cargo de toda la funcionalidad de Tiempo Real, incluyendo interrupciones, planificación de procesos y temporización de alta resolución. Este micro kernel corre el kernel estándar como una tarea en segundo plano.

•Kernel IEEE 1003.1d: Esta segunda aproximación consiste en implementar las extensiones para Tiempo Real de POSIX.1 dentro de la estructura del kernel Linux estándar. Estas extensiones agregan un temporizador, la planificación y la lógica para apropiación de procesos dentro de un único kernel monolítico.

Para ambos casos, las modificaciones del kernel Linux se distribuyen como parches al kernel estándar, es decir que ninguna de estas variantes de Linux para Tiempo Real son partes del código fuente oficial del kernel Linux distribuido en kernel.org. Dos ejemplos de implementación de micro kernel son RTLinux, y RTAI. Con extenciones Posix está Linux Preempt, KURT (The Kansas University Real Time Linux), TimeSys Linux, MaRTEOS. Una variante similar al microkernel es el ADEOS, que implementa el nanokernel.

3 Experimentación: Sistemas Operativos Evaluados

En las pruebas realizadas se tomaron mediciones de la latencia del planificador de kernel para los siguientes Sistemas Operativos:

- RTAI – Knoppix v1.2. Ver. Kernel Linux: 2.6.17.11 [3] [11]
- RTLinux STRTL v1.0.Ver. Kernel Linux: 2.4.29-rtl-3.1 [7] [8] [14] [16] [17]
- Linux kernel estándar Ubuntu v10.04 LTS. Ver. Kernel Linux: 2.6.32-21-generic
- Linux con parche RT-Preempt OSADL. Ver. Kernel Linux: 2.6.31-rt10 [9] [13] [15]

3.1 Sistema de Pruebas

Todas las pruebas se realizaron sobre un único sistema de cómputo, con las siguientes características:

- Arquitectura CPU: Intel x86
- Procesador: Intel Pentium III Coppermine 733 Mhz
- Frecuencia de bus externo: 133 MHz

- Tamaño de cache: 256K
- Memoria del sistema: 512 MB

3.2 Objetivo y metodología de las pruebas

El objetivo de las pruebas realizadas consistió en evaluar el desempeño en relación al tiempo de respuesta para los diferentes sistemas operativos. Para ello, se tomó como métrica principal la latencia en el **peor caso** [2] [18]. Normalmente, la latencia máxima de un sistema se mide usando un interruptor externo que introduce eventos al sistema mediante interrupciones. Por ejemplo, para hacer esto podría usarse la línea de handshake de un puerto serie o paralelo. Este método sin embargo requiere instrumental especializado y circuitos externos. Para medir la latencia máxima, se utilizaron las herramientas provistas por los desarrolladores de los sistemas operativos de Tiempo Real. En general, estas pruebas consisten en el inicio reiterado de tareas de alta prioridad, tomando la cuenta de ciclos de CPU para determinar el tiempo transcurrido entre que se produce el evento y el instante en el que comienza a ser atendida. Para cada caso, las pruebas se realizaron tanto bajo condiciones de CPU ociosa como en condiciones de carga. Se escribieron scripts y se utilizaron programas que permitieran sobrecargar al sistema en el uso de CPU, en la generación de interrupciones y en la atención de eventos de disco y red. También se construyeron los scripts necesarios para compilar dichas herramientas, generar la carga necesaria en el sistema, y se especificaron los parámetros que permitan controlar su ejecución, de forma que las pruebas en los diversos sistemas sean consistentes. Con la metodología utilizada se mide la latencia de planificación, que representa la porción final de la latencia total. La latencia de hardware y la generada al ejecutar la rutina del servicio de interrupción no se miden. De todas formas, los tiempos de estas latencias no suelen ser significativos en comparación a las latencias de planificación de los sistemas operativos, y, además, este trabajo es para evaluar SOTR.

4 Resultados

4.1 RTAI

En RTAI [3] [11], sin carga de CPU, las latencias máximas son bajas (entre 2000 y 4000 ns), con escasos picos de alrededor de 200 microsegundos. En el gráfico de distribución, la mayor parte de las muestras de latencia obtenidas se concentran entre los 1400 y 4000 nanosegundos. Podemos ver, cotejando este gráfico con el de latencias que la fluctuación se correlaciona directamente con la latencia. La fluctuación, en promedio es de 3091,34615 nanosegundos.

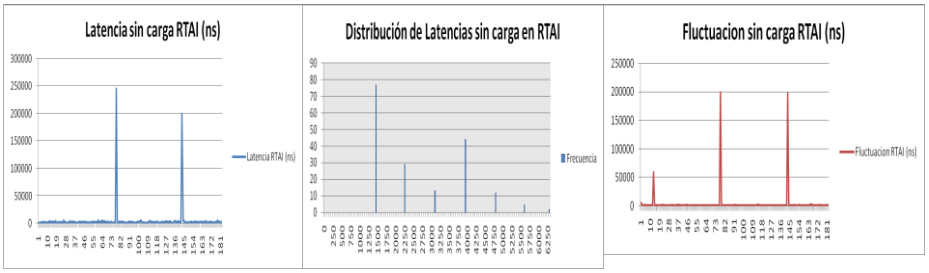

Fig.2: a) Latencia RTAI sin carga b) Gráfico de distribución de latencias c) Fluctuación sin carga

Puede observarse que con la sobrecarga de CPU los valores de latencia máxima se han incrementado. Se encuentran en esta muestra una cantidad mayor de valores que superan los 6000ns, pero no se encuentran incrementos importantes en la latencia máxima para la tarea de Tiempo Real en el kernel cuando se sobrecarga la CPU. Esto también se puede apreciar en el gráfico de distribución. La mayor parte de las muestras de latencia máxima obtenidas bajo carga de CPU en RTAI se encuentran entre los 3800 y 8000 nanosegundos. El valor promedio de latencia máxima obtenido es de 5586,5 nanosegundos. El valor de fluctuación promedio es de 3224 nanosegundos. Se mantiene la correlación con la latencia máxima, y no se notó un cambio importante en los valores respecto al caso de CPU ociosa.

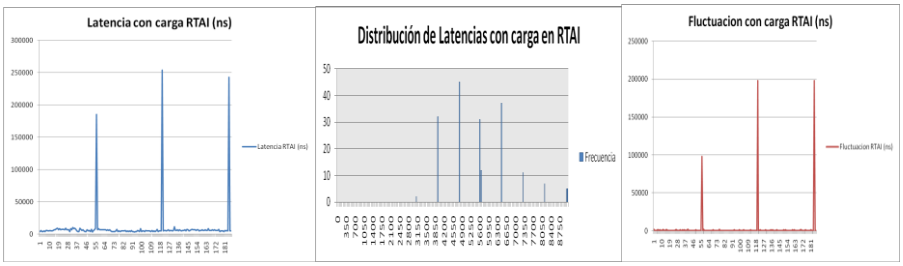

Fig.3: a) Latencia RTAI con carga b) Gráfico de distribución de latencias c) Fluctuación con carga

Nota sobre valores pico de latencia máxima: Los valores pico que pueden verse en el gráfico de latencias máximas son causados por condiciones específicas del sistema, conocidas como condiciones "mata latencia" ("latency killers") que suelen ser generadas por diversas causas, tales como el soporte del puerto USB, el sistema administrador de energía APM o ACPI, o el inicio de interrupciones SMI de modo de administración de sistema SMM, entre muchas otras. Para el caso de una implementación de Tiempo Real con restricciones de tiempo concretas, estos valores altos en la medición de latencia pueden no ser aceptables, y se suele cambiar la configuración en el BIOS o la configuración kernel del sistema operativo para evitar dichas condiciones. Para el propósito de estas pruebas consideramos que son tolerables, ya que solo ocurren ocasionalmente y son claramente identificables. Excluyendo los valores de latencia debidos a estas condiciones, tenemos que la latencia promedio es de 2620,4 nanosegundos (2620 * 10-9 = 0,000002620 segundos).

4.2 RTLinux

En RTLinux [7] [8] [14] [16] [17], sin carga de CPU, las latencias máximas obtenidas en RTLinux para el sistema de pruebas son similares a las obtenidas en RTAI. Con excepción de los picos ocasionales debidos al trabajo del planificador, las latencias en RTAI en condiciones de CPU ociosa se encuentran entre los 1200 y 2400 nanosegundos, como puede verse en el gráfico de distribución. Los valores de latencia superiores a los 9000 nanosegundos son pocos y esporádicos, en relación al total de las muestras. El valor de latencia máxima promedio obtenido es de 1637,7 nanosegundos.

Con carga de CPU, al igual que en el caso de RTAI, puede notarse en RTLinux un efecto mínimo de la carga de CPU sobre la latencia del planificador. Si bien una gran parte de las muestras tienen valores de latencia similares a aquellos obtenidos con la CPU en estado ocioso, puede verse también que hay un conjunto de muestras con valores entre los 7800 y 13200 nanosegundos.

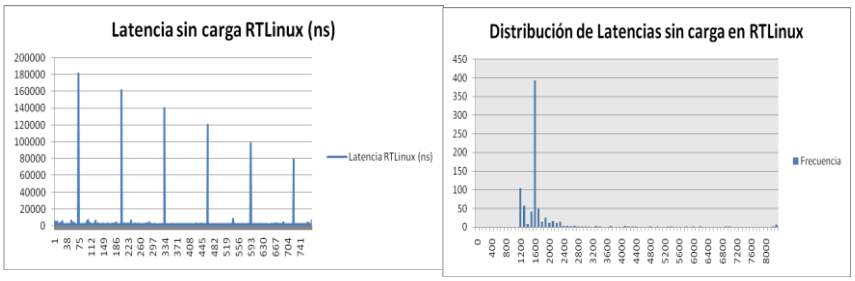

Fig.4: a) Latencia RTLinux sin carga b) Gráfico de distribución de latencias

Aún así, los valores de latencia obtenidos (en promedio de 3538,7 nanosegundos) son muy bajos en comparación a los valores obtenidos en una distribución con kernel estándar de Linux.

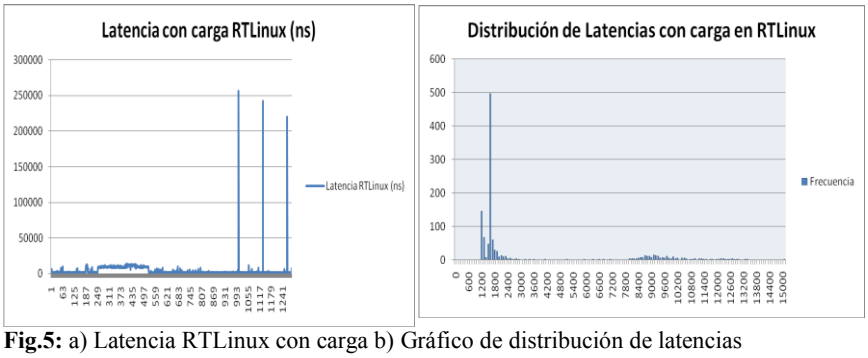

Fig.5: a) Latencia RTLinux con carga b) Gráfico de distribución de latencias

4.3 Resultados Linux con Kernel Estándar

Las mediciones de latencia obtenidas para un kernel Linux estándar, incluso bajo condiciones de CPU ociosa, están en el orden de los microsegundos (10-6 segundos), al contrario que en los sistemas operativos de Tiempo Real evaluados, cuyas latencias

máximas se encontraron generalmente en el orden de los nanosegundos (10-9 segundos). En el gráfico de distribución se recolectan los datos tomados, y puede verse que en gran medida las latencias obtenidas en un kernel estándar Linux con baja carga de CPU están entre los 24 y 40 microsegundos (24000 y 40000 nanosegundos). El valor promedio de latencia máxima en ausencia de carga es de 32,14 microsegundos.

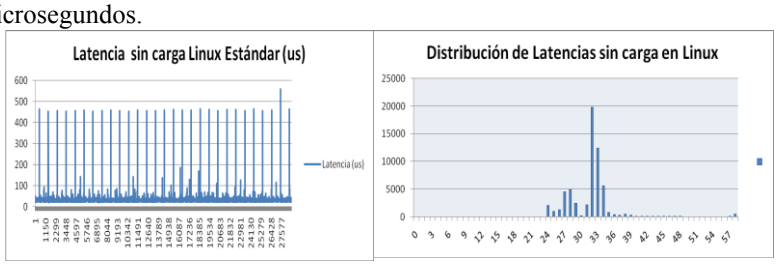

Fig.6: a) Latencia Linux Estándar sin carga b) Gráfico de distribución de latencias

Con carga de CPU, en el gráfico se toma un fragmento del total de las muestras, y se ve que las medidas de latencia para el kernel Linux estándar se incrementaron notablemente bajo una situación de carga de CPU. El siguiente gráfico de distribución abarca todos los datos tomados, y se puede ver que una gran cantidad valores de latencia están entre el millón y los 4,5 millones de microsegundos, es decir, entre 1 y 4,5 segundos. En términos de tiempos de planificación de CPU, los valores obtenidos en este caso son extremadamente altos. El gráfico de distribución abarca todos los datos tomados, y se puede ver que una gran cantidad valores de latencia están entre el millón y los 4,5 millones de microsegundos, es decir, entre 1 y 4,5 segundos. En términos de tiempos de planificación de CPU, los valores obtenidos en este caso son extremadamente altos.

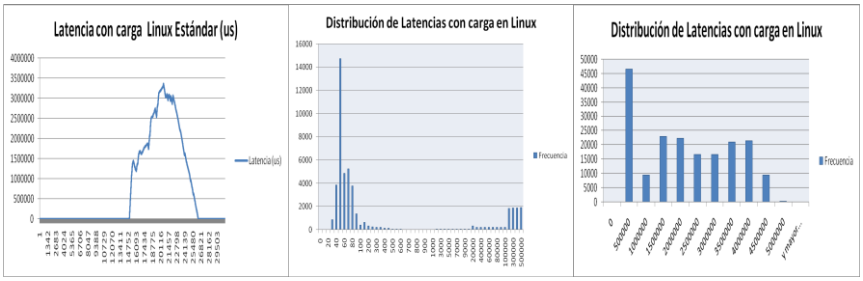

Fig.7: a) Latencia Linux Estándar c/carga b) Hist. de latencias c)Gráfico de distribución detallado

Aún así, hay una cantidad apreciable de muestras entre 0 y 500000 microsegundos, las cuales se analizan en el gráfico de distribución. Puede apreciarse que en este sub-rango existe una cantidad las latencias entre los 20 y 600 microsegundos, valores similares a los obtenidos en ausencia de carga, pero en general la proporción total de las latencias obtenidas bajo carga de CPU fueron mayores a los 100000 microsegundos. En promedio, el valor de latencia obtenido es de 1873405,6 microsegundos (1837405597 nanosegundos).

4.4 Resultados Linux con Kernel Apropiativo (con parche RT-Preempt)

En Linux con parche RT-Preempt [9] [13] [15], el siguiente gráfico muestra las latencias sin carga de CPU para un proceso de alta prioridad obtenidas en un kernel Linux al cual se le ha aplicado el parche RT-Preempt: En ausencia de carga de CPU, los valores obtenidos son similares a aquellos obtenidos en el kernel estándar bajo las mismas condiciones.

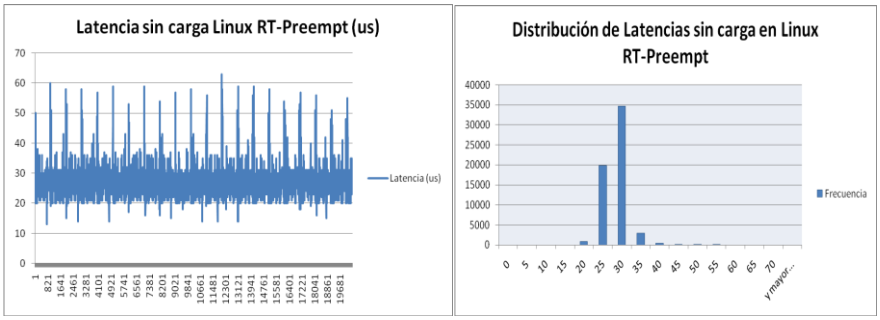

Fig.7: a) Latencia Linux RT-Preempt sin carga b) Gráfico de distribución de latencias

La gran mayoría de los valores de latencia máxima obtenidos se encuentran entre los 20 y 45 microsegundos (20000 y 45000 nanosegundos), y el valor promedio es de 26,9 microsegundos.

Los beneficios de aplicar el parche para hacer apropiativo al kernel Linux se aprecian notablemente bajo condiciones de carga de CPU. En comparación con los valores obtenidos para kernel estándar sin el parche, las latencias para el proceso de alta prioridad se redujeron a valores cercanos a aquellos obtenidos bajo ausencia de carga. Como se ve en la distribución, los valores de latencia obtenidos varían entre los 25 y 64 microsegundos, y se encuentran en mayor medida entre los 32 y 44 microsegundos. Del total de las muestras tomadas, el valor promedio de latencias máximas es de 38,94 microsegundos.

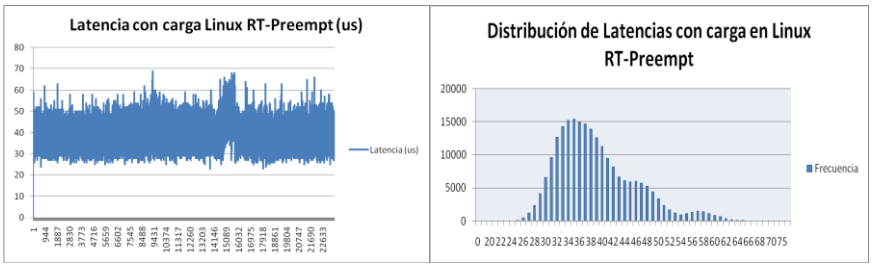

Fig.7: a) Latencia Linux Estándar con carga b) Gráfico de distribución de latencias

5 Conclusión

El propósito principal de este trabajo consistió en verificar el desempeño de los SOTR libres basados en Linux, y comparar los resultados obtenidos en dichos

sistemas con otros basados en el kernel Linux estándar. Para ello, en vez de recurrir directamente a las mediciones decidimos partir desde una base teórica que nos permitiese conocer con mayor profundidad los sistemas operativos evaluados y en qué consistirían realmente los datos a medir. Habiendo revisado los conceptos teóricos necesarios, la realización de las pruebas nos permitió constatar empíricamente las mejoras en tiempo de respuesta obtenidas por los sistemas libres de Tiempo Real basados en Linux considerados. En relación con la distribución estándar del kernel Linux, pudimos verificar que en promedio, bajo condiciones de carga, las latencias obtenidas RTAI son 320 mil veces más bajas. Además, pudimos verificar que en comparación con las latencias obtenidas en RTLinux bajo las mismas condiciones, las latencias que se midieron en el kernel estándar son 520 mil veces mayores. Esto significa que los dos sistemas operativos de Tiempo Real con arquitectura micro kernel evaluados, se obtuvieron latencias en el orden de los cientos de miles de veces más bajas que en el kernel estándar. Adicionalmente, los resultados obtenidos en el kernel Linux con el parche RT-Preempt muestran que se obtienen mejoras significativas en las latencias respecto al kernel estándar sin el parche, ya que los tiempo obtenidos son, en promedio, 47 mil veces más bajos. Los valores obtenidos explican el hecho de que esta modificación del kernel Linux permite obtener un rendimiento óptimo en aplicaciones que requieren un tiempo de respuesta en el orden de los cientos de microsegundos o en el orden de los milisegundos. Sin embargo, como pudimos comprobar, la aplicación del parche RT-Preempt no es suficiente para garantizar la respuesta a tiempo a aquellas aplicaciones de Tiempo Real estricto que las requieren en el orden de unos pocos miles de nanosegundos. Por otra parte, los resultados muestran que los valores promedio de latencia logrados mediante la aplicación del parche RT-Preempt son, por lo menos, 7 veces mayores que los valores obtenidos en los sistemas micro kernel. En resumen, podemos concluir que los resultados de las pruebas confirman el motivo por el cual tanto RTAI como RTLinux son sistemas ampliamente utilizados en entornos y aplicaciones que demandan tiempos de respuesta mínimos, tales como la ingeniería aeroespacial, los procesos de fabricación industrial, la robótica o instrumentales de alta precisión. Por otro lado, si bien los tiempos de latencia alcanzados por el parche RT-Preempt no son tan bajos como los obtenidos en los sistemas micro kernel, permiten reducir considerablemente los tiempos obtenidos en un kernel Linux estándar, y por ello es recomendable el uso de este parche para aplicaciones de Tiempo Real que no sean críticas, pero que requieran una mejora en los tiempos de respuesta.

Bibliografía

1. N.C. Audsley , A. Burns , M. F. Richardson , A. J. Wellings: Hard Real-Time Scheduling: The Deadline-Monotonic Approach. Proc. IEEE Workshop on Real-Time Operating Systems and Software (1991).

2. Barba- lace, A.; Luchetta, A.; Manduchi, G.; Moro, M.; Soppelsa, A.; Taliercio, C.: Performance Comparison of VxWorks, Linux, RTAI, and Xenomai in a Hard Real Time Application. Real-Time Conference, 15th IEEE-NPSS, (2007).

3. Dave Beal: RTAI (Real Time Application Interface). http://www.linuxfordevices. com/c/a/Linux- For-Devices-Articles/RTAI-RealTime-Application-Interface/ (2000)

4. L. Buhr. “An Introduction to Real Time Systems”. Prentice Hall (1999).

5. A. Burns, A. J. Wellings: Designing hard real-time systems. Proceedings of the 11th Ada-Europe international conference on Ada. ISBN:0-387-55585-4 (1992).

6. A. Burns & A. Wellings: Real-Time Systems and Programming Languages. Addison Wesley, ISBN 90-201-40365-x

7. FSM Labs, Inc.: Getting Started with RTLinux. http://www.ee.nmt.edu/~rison/ ee352_fall09/Getting_Started_with_RTLinux.pdf (2001).

8. Nicholas Mc Guire: RTLinux/GPL Kickstart, Lanzhou University http://dslab.lzu.edu.cn:8080/members/hofrat/kickstart_rtl.pdf (2007)

9. OSADL: Open Source Automation Development Lab https://www.osadl.org/

10. Frederic Proctor: Introduction to Linux for Real-Time Control. National Institute of Standards and Technology. http://www.aeolean.com/html/ RealTimeLinux/RealTimeLinuxReport-2.0.0.pdf (2002)

11. Giovanni Racciu, Paolo Mantegazza: RTAI 3.4 User Manual. https://www.rtai.org/ index.php?module=documents &JAS_DocumentManager_op=downloadFile&JAS_ File_id=46 (2006)

12. Ismael Ripoll, Pavel Pisa, Luca Abeni, Paolo Gai, Agnes Lanusse, Sergio Saez: RTOS State of the Art Analysis. DISCA, Universidad Politecnica de Valencia, http://www.mnis.fr/ocera_support/rtos/ (2002).

13. Real-Time Linux: RT Preempt Kernel Patch https://rt.wiki.kernel.org/ index.php/Main_Page

14. RTLinux Portal at Universidad Politécnica de Valencia. http://rtportal.upv.es/

15. Siro Arthur. Carsten Emde. Nicholas Mc Guire: Assessment of the Realtime Preemption Patches (RT-Preempt) and their impact on the general purpose performance of the system. 9th RTL Workshop (2007).

16. Phil Whilshire, Kaiser Aluminum, Victor Yodaiken, Joachim Nilsson, Daniel Rytterlund: RTLinux Installation Manual. Real Time Linux Documentation Project, Vol. 1, http://vmlinux.org/rtl/contrib/ installation/ Installation_Manual.pdf (2000)

17. Victor Yodaiken: An Introduction to RTLinux. http://www. Linuxfor devices. com/c/a/Linux-For-Devices-Articles/An-Introduction-to-RTLinux/ (1999).

18. Diego Lopez Zamarrón: Análisis de Sistemas Operativos de Tiempo Real. http://gayuba1.datsi. fi.upm.es/~dlopez/ (2004).