Planificador combinado: una estrategia para mejorar el rendimiento de los sistemas de tiempo real crítico

José María Aceituno, Ana Guasque, Patricia Balbastre, José Simó, Alfons Crespo
Instituto de Automática e Informática Industrial (ai2)
Universitat Politècnica de València

Resumen

En este trabajo se propone y evalúa una nueva política de planificación en sistemas multinúcleo de tiempo real. La planificación combinada trata de establecer una nueva estrategia de planificación que reduzca los cambios de contexto y las interferencias generadas con el uso de recursos compartidos entre los núcleos. Esta estrategia se basa en la idea de elegir la mejor política para un determinado intervalo de ejecución durante todo el plan. El resultado es una planificación de tareas implementada con diferentes políticas como EDF (Earliest Deadline First) o DM (Deadline Monotonic) y variantes de las mismas. Se ha evaluado la propuesta con diferente número de núcleos y carga obteniendo mejoras con respecto al uso de un solo planificador en todo el horizonte de ejecución.

Palabras clave: sistemas de tiempo real, planificación, cambios de contexto, multinúcleo.

1. Introducción

El uso de sistemas empotrados está actualmente muy extendido, no sólo en el sector industrial, sino en todos los aspectos de la vida moderna. La potencia de procesamiento de los sistemas multinúcleo permite utilizar múltiples aplicaciones embebidas en una única plataforma de hardware compartida.

La teoría existente en el campo de los sistemas multinúcleo muestra que la planificación en dichos sistemas es compleja (NP-Hard). La generación de planes cíclicos a partir de los requerimientos temporales de varias aplicaciones en un sistema multinúcleo requiere el uso de técnicas y heurísticas que intenten conseguir planificaciones factibles en un tiempo limitado. Un gran número de elementos adicionales como la diferente criticidad de las aplicaciones, la asignación de tareas a los núcleos, la gestión del consumo energético, la optimización del rendimiento del sistema operativo, etc., añaden complejidad y dificultad a la generación de planificaciones factibles y eficientes.

Pero no sólo existe la dificultad de generar un plan factible en los sistemas multinúcleo, también es difícil estimar el tiempo de cómputo de peor caso de las tareas. Un problema fundamental para estimar los tiempos de cómputo de las tareas es la falta de información precisa sobre el tiempo de algunas instrucciones en todas las circunstancias debido a las complejas dependencias de las instrucciones y las interacciones de la memoria.

La determinación precisa del WCET (Worst Case Execution Time) es una tarea difícil incluso en arquitecturas de un solo núcleo. Los problemas se acentúan en el contexto multinúcleo principalmente debido a la compartición de recursos que puede dar lugar a interacciones muy complejas y a indeterminismo.

Desde el punto de vista de la plataforma de ejecución, algunos parámetros de sobrecarga asociados al sistema operativo, como el cambio de contexto de las tareas, también deben ser considerados para la generación del plan de ejecución debido al nivel de sobrecarga que pueden introducir. Al generar el plan de ejecución, al tiempo de cálculo efectivo requerido para la ejecución de las actividades internas de la tarea, debe añadirse el tiempo de cambio de contexto.

El objetivo de este trabajo es buscar alternativas a los planificadores de sistemas multinúcleo de tiempo real para que, no solo se obtenga un plan planificable, sino que se intente minimizar algunos parámetros significativos del sistema. En concreto, se propone un nuevo tipo de planificador (llamado planificador combinado) que obtenga un plan planificable y que tenga como objetivos adicionales reducir interferencias y cambios de contexto.

Nos centramos en el ámbito de los sistemas de tiempo real crítico.

2. Trabajos relacionados

Existen muchos trabajos sobre planificación en sistemas multiprocesador. El *survey* de [3] es de obligada lectura en esta materia. Con respecto al estudio de las interferencias producidas por el acceso a recursos *hardware* compartidos, en [10] se describen los trabajos más importantes hasta 2021. La

 https://doi.org/10.17979/spudc.9788497498418.0870

mayoría de trabajos que tratan de reducir esta interferencia, lo hacen para algún tipo específico de recurso hardware: bus de memoria ([2], [8]), memoria DRAM [7]. En nuestro caso, adoptamos un modelo general para cualquier tipo de recurso de forma similar a como se hace en [1] y [4].

Algunos trabajos tratan también de reducir algunos parámetros temporales utilizando técnicas de planificación no convencionales como la programación lineal entera. En [6] se utiliza esta técnica en sistemas mono núcleo para obtener planificaciones que minimizan cambios de contexto o tiempos de respuesta de peor caso.

3. Modelo de tareas

Suponemos un sistema multinúcleo con m núcleos $(M_0, M_1, M_2, ..., M_{m-1})$ donde se debe asignar un conjunto de tareas τ de n tareas independientes. Cada tarea τ_i está representada por la tupla

$$\tau_i = (C_i, D_i, T_i, I_i) \quad (1)$$

donde C_i es el tiempo de cómputo de peor caso (WCET), D_i es el plazo relativo, T_i es el período y I_i es la interferencia. El término I_i es el tiempo que tarda la tarea en acceder a los recursos de hardware compartidos. Un caso típico es la operación de lectura y escritura en memoria. Aunque I_i forma parte de C_i, durante el tiempo que la tarea accede al recurso compartido, otras tareas en otros núcleos experimentarán retardos si intentan acceder al mismo recurso. Así que este tiempo de interferencia se define independientemente de C_i, ya que se utilizará para representar el retraso causado a otras tareas. Una descripción detallada de este parámetro se puede encontrar en [1]. Asumimos plazos limitados, por lo que $D_i \leq T_i$. Las tareas pueden ser periódicas o esporádicas. Este modelo de tareas es el mismo que el presentado en [5].

4. Planificación de sistemas de tiempo real crítico multinúcleo

En el ámbito de la planificación de sistemas de tiempo real crítico multinúcleo, para obtener un plan planificable, se pueden identificar dos pasos:

1. Asignación de tareas a núcleos: En este paso se suelen utilizar algoritmos de bin-packing de forma general, o algoritmos que busquen minimizar algún parámetro temporal. En concreto, respecto a algoritmos de bin-paking, los más usuales son:
   - FFDU (First Fit Decreased Utilization). Asigna una tarea al primer procesador donde es posible. El orden de elección de las tareas es por utilización decreciente.
   - WFDU (Worst Fit Decreased Utilization). Asigna una tarea al procesador que tiene el menor hueco de utilización disponible. El orden de elección de las tareas es por utilización decreciente.

   Con respecto a algoritmos de asignación que buscan, no solo alojar las tareas sino minimizar algun parametro temporal, destacamos el algoritmo Wmin [1] que, mediante programación lineal entera, obtiene la asignación óptima para minimizar la interferencia total generada por el uso de recursos hardware compartidos.
2. Planificación mono núcleo: Una vez se asignan las tareas a los núcleos, ya que no se contempla la migración (por ser un sistema de tiempo real crítico), sólo queda planificar cada núcleo con la política de planificación para mono procesador elegida.

4.1. Intervalos de ejecución

Tal y como se define en [11], un intervalo de ejecución i $[t_i^s, t_i^e)$ para un conjunto de tareas se define como aquellos intervalos en los que no se produce ningún instante ocioso en el procesador. En [11] se puede consultar la fórmula para calcular cada intervalo.

De este modo, la ejecución a lo largo del hiperperiodo del conjunto de tareas es una sucesión de intervalos de ejecución e intervalos ociosos.

En la siguiente sección se propone un planificador que utiliza un algoritmo de planificación diferente en cada intervalo de ejecución.

5. Planificador combinado

Tal y como se introduce anteriormente, el planificador combinado (referido en este artículo a veces como P.C.) es un algoritmo de planificación que utiliza o combina distintas políticas de planificación como pueden ser EDF o DM[9] en un mismo plan. La clave del P.C. es que realiza la planificación intervalo por intervalo, y en cada uno de ellos puede aplicar una política de planificación diferente, es decir, la que mejor resultados genere para dicho intervalo para ciertos parámetros temporales. Esto lo realiza de forma totalmente independiente del resultado que esa misma política pudiera tener en los otros intervalos.

Los intervalos de ejecución están delimitados por

https://doi.org/10.17979/spudc.9788497498418.0870

momentos ociosos de la CPU. En un sistema multinúcleo consideramos un momento ocioso cuando todos y cada uno de los núcleos del sistema estén ociosos, es decir, no tengan ninguna tarea en ejecución. De esa manera aseguramos que el cambio de política de un intervalo no afecta al anterior intervalo o al posterior, dado que un instante ocioso en todos los núcleos significa que no hay tareas pendientes de ejecutar en ese instante.

En el planificador combinado que hemos desarrollado hemos considerado hasta 6 políticas de planificación distintas que son las siguientes: EDF original, variante 1 de EDF, variante 2 de EDF, DM original, variante 1 de DM y variante 2 de DM. Los EDF y DM originales, son las mismas políticas EDF y DM que se usan típicamente en planificación. Por otro lado, las dos variantes propuestas de cada planificador tradicional son dos optimizaciones cuyo objetivo es la reducción de cambios de contexto. A continuación se explica cada variante y se ilustra con un ejemplo.

5.1. Variante 1

La variante 1 (aplicable tanto a EDF como DM) consiste en no expulsar a una tarea que está en ejecución cuando otra tarea más prioritaria se activa. No se realizaría expulsión si el tiempo que le resta a la tarea en ejecución en terminar es menor que el tiempo de cómputo de la tarea más prioritaria. En este caso, la tarea en ejecución no sería expulsada, herendando la prioridad de la tarea que la quiere expulsar.

Supongamos el conjunto de 3 tareas: $\tau_0 = (4, 25, 25, 0)$, $\tau_1 = (5, 30, 30, 0)$ y $\tau_2 = (30, 100, 100, 0)$ en un mismo núcleo, bajo la política EDF. Según la política EDF, en el instante 240 τ_1 tiene mayor prioridad que τ_2, por tanto, según EDF original (también referido en este artículo como EDF 0) en el instante 240, que es cuando la tarea τ_1 es activada, esta expulsará de la CPU a la tarea τ_2 y hasta que no acabe su ejecución τ_2 no podrá volver a ocupar la CPU. Este comportamiento puede observarse en la parte superior de la Figura 1. Sin embargo, con la aplicación de la variante 1, antes de la expulsión, se comprobaría cuántas unidades de tiempo (u.t.) restan a τ_2 para finalizar su ejecución, que en este caso serían 3, con las unidades de tiempo que necesita τ_1 para ejecutarse, que en este caso serían 5, todo su WCET. Dado que τ_1 necesita más tiempo que τ_2 para finalizar, τ_2 heredaría la prioridad de τ_1 y por tanto permanecería en la CPU hasta finalizar su ejecución. Este comportamiento puede visualizarse en la parte inferior de la Figura 1. Este mismo fenómeno que produce la variante 1 resultaría igual si lo aplicásemos a la política DM.

El objetivo de esta variante es reducir los cambios de contexto.

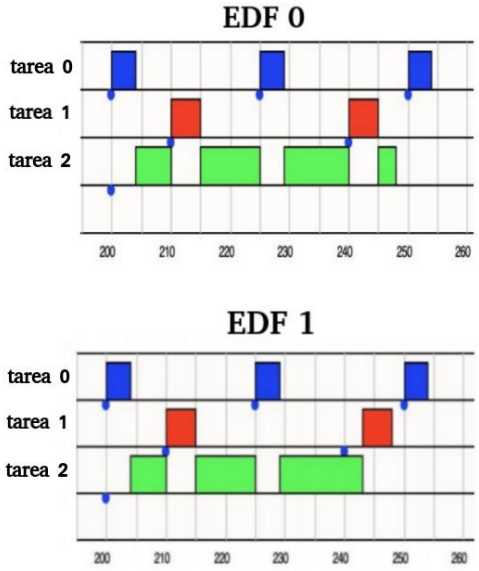

Figura 1: Ejemplo de la variante 1

5.2. Variante 2

La variante 2 consiste en la declaración de una tarea como no expulsiva durante N unidades de tiempo cuando dicha tarea es asignada a un núcleo, ya sea el caso que la tarea va a iniciar su ejecución o ya sea que la tarea hubiera sido expulsada anteriormente. Esta variante puede entenderse como que las tareas tienen un tiempo de bloqueo al inicio de ser asignadas a la CPU durante el cual no pueden ser expulsadas. El valor de N puede establecerse en función de la cantidad de tareas, la duración de las mismas o la utilización del sistema. En cualquier caso esta variante es aplicable tanto a EDF como a DM y su objetivo también es reducir el número de cambios de contexto.

Supongamos un sistema de un núcleo con 4 tareas: $\tau_0 = (5, 22, 22, 0)$, $\tau_1 = (6, 25, 25, 0)$, $\tau_2 = (4, 28, 28, 0)$ y $\tau_2 = (10, 100, 100, 0)$, bajo la política de EDF original. En el instante 140, la tarea τ_2 tiene más prioridad que la tarea τ_3, sin embargo, tal y como se muestra en la Figura 2, bajo la política de EDF 2 con N=10, la tarea τ_2 y la tarea τ_3 no sufrirían ninguna expulsión. Esto puede observarse comparando la parte inferior de la Figura 2 donde no ocurre ninguna expulsión durante el intervalo de planificación mostrado, al contrario que con EDF original (EDF 0) donde las tareas τ_2 y τ_3 sufren varias expulsiones. De este modo, en este ejemplo y durante el intervalo mostrado, la variante 2 consigue eliminar hasta 4 cambios de contexto.

https://doi.org/10.17979/spudc.9788497498418.0870

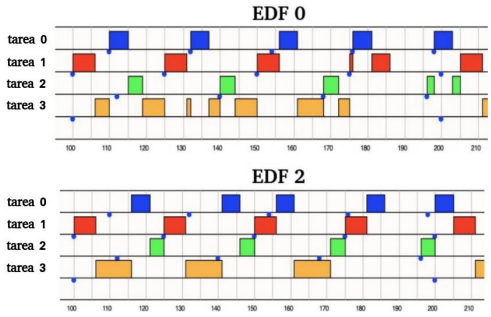

Figura 2: Ejemplo de la variante 2

5.3. Funcionamiento del planificador combinado

Una vez que están claras las 6 posibles políticas de planificación a elegir por el planificador combinado (EDF0, EDF1, EDF2, DM0, DM1, DM2), vamos a explicar con un poco más de detalle cómo funciona el mismo.

El objetivo del planificador es dual: reducir cambios de contexto y/o reducir interferencia por acceso a recursos hardware compartidos. En base a esto, tendremos dos versiones del P.C.:

- P.C. cambios de contexto: tiene como objetivo prioritario reducir cambios de contexto y en ello basa sus decisiones sobre cual variante utilizar en cada intervalo de ejecución.
- P.C. interferencia: tiene como objetivo prioritario reducir interferencia por acceso a recursos hardware compartidos y en ello basa sus decisiones sobre cual variante utilizar en cada intervalo de ejecución.

El planificador combinado actúa de manera idéntica en cada intervalo. Para cada intervalo el planificador combinado simula el resultado que se obtendría si fuese planificado por cada una de las políticas de planificación disponibles, en nuestro caso, con las 6 políticas(EDF0, EDF1, EDF2, DM0, DM1, DM2). Tras planificar con cada una de esas políticas se obtendrá una cantidad de interferencia determinada, un número de cambios de contexto determinados y ante todo, si es planificable o no esa política para dicho intervalo en ese conjunto de tareas. A partir de toda esta información, el planificador combinado escogerá la política de planificación que le proporcione mejores resultados en función de la versión utilizada.

El P.C. siempre elegirá en cada intervalo el planificador que mejor valor obtenga según el objetivo prioritario, pero en caso que haya un empate en los resultados entre dos o más políticas, se comprobará el resultado según el otro objetivo para decidir cuál política es la escogida. Obviamente las políticas que no sean capaces de planificar el intervalo directamente serán descartadas. En caso de que un planificador obtenga los mismos resultados para ambos criterios se escogerá uno por defecto.

5.4. Ejemplo de ejecución del Planificador Combinado

Para entender mejor el planificador combinado pondremos un ejemplo. Supongamos un sistema de dos núcleos, con 4 tareas: τ_0, τ_1, τ_2 y τ_3 con las siguientes propiedades: $\tau_0 = (2, 6, 6, 1)$, $\tau_1 = (3, 10, 10, 0)$, $\tau_2 = (2, 6, 6, 0)$ y $\tau_3 = (3, 8, 8, 1)$.

Para simplificar el ejemplo, vamos a suponer que el planificador combinado solo puede elegir entre 2 políticas disponibles, EDF0 y EDF1. Utilizaremos la versión que minimiza los cambios de contexto.

Como primer paso, el planificador combinado planificará el conjunto de tareas con EDF0 para el primer intervalo de ejecución. Como puede observarse en la Figura 3 obtiene un intervalo corto, de solo 5 u.t., desde el instante 0 al instante 5. El resultado de esa planificación de intervalo es de: planificable, 0 cambios de contexto y 0 de interferencia. De ese modo, el P.C. volvería al instante 0 y volvería a planificar con EDF1 el mismo intervalo. En este caso obtendrá el mismo resultado con EDF1 que con EDF0.

Al tener todas las opciones el mismo resultado, el P.C. se quedará con la política asignada que tenga como predeterminada, supongamos en este caso que fuese EDF0.

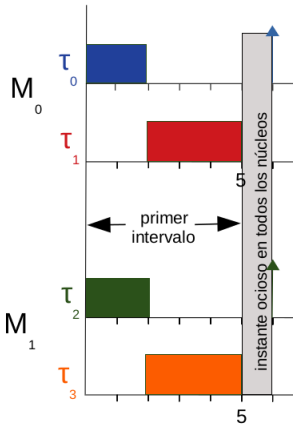

Figura 3: Planificación resultante tras planificar un intervalo con EDF0 (el resultado sería el mismo que con EDF1)

Tras haber planificado el primer intervalo, el P.C. se situaría en el final del primer intervalo, en este caso en el instante 6 y por tanto, el P.C. volvería

a repetir el proceso.

El P.C. volvería a planificar con EDF 0 desde el instante 6 hasta el siguiente momento ocioso, que tal y como puede observarse en la Figura 4 sería el instante 15. Con esta opción, se obtendría planificabilidad, 1 cambio de contexto (de la tarea 1) y 0 de interferencia. El P.C. registraría dicho resultado, retornaría al instante 6 y volvería a planificar, esta vez, con la otra política, EDF1.

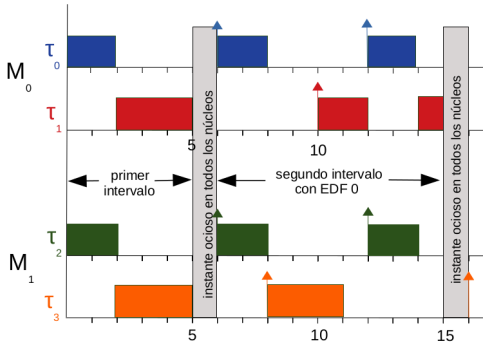

Figura 4: Planificación resultante tras dos intervalos usando en ambos EDF0

Tal y como puede observarse en la Figura 5 el resultado de planificar el segundo intervalo con EDF1 sería distinto: se obtendría planificabilidad, 0 u.t. de interferencia y 0 cambios de contexto, lo cual sería un mejor resultado que la anterior opción, la de planificar con EDF0.

Por tanto, el P.C. elegiría EDF 1 como la política adecuada para el segundo intervalo, de ese modo pasaría a resolver el siguiente intervalo de la planificación y así sucesivamente hasta llegar al final.

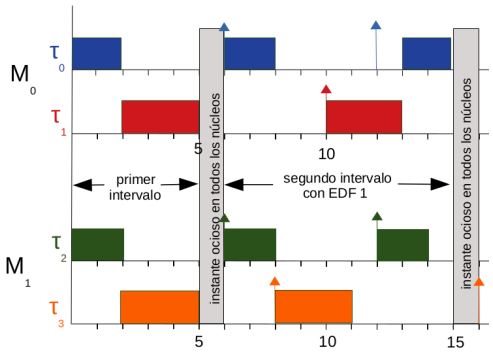

Figura 5: Planificación resultante tras planificar el segundo intervalo con EDF1

6. Evaluación

Para evaluar los resultados del planificador combinado, hemos realizado simulaciones experimentales en las cuales creamos conjuntos de tareas de modo aleatorio, con hiperperiodos aleatorios de hasta 3000 u.t.. Dichos conjuntos son planificados de manera íntegra por varios planificadores: el P.C. en su versión de minimizar la interferencia, el P.C. en su versión de minimizar los cambios de contexto, EDF 0 (EDF original), EDF 1, EDF 2, DM 0 (DM original) , DM 1 y DM 2.

Los conjunto de tareas se han creado para 4 escenarios distintos, con diferente cantidad de núcleos, de modo que la utilización del sistema y el número de tareas en cada escenario va acorde al número de núcleos del sistema simulado tal y como se presenta en el Cuadro 1.

Además, para una evaluación más completa, hemos realizado la simulación experimental 3 veces, cada una con un asignador de tareas distinto: WFDU , FFDU y Wmin. Para la evaluación de los distintos planificadores, hemos establecido 3 criterios claves a comparar: la planificabilidad, la interferencia producida y los cambios de contexto.

A continuación, detallamos los resultados obtenidos de todas las simulaciones experimentales mostrando los resultados en gráficas promedio, es decir, gráficas que muestran la media de los 4 escenarios descritos en el Cuadro 1. Por cada escenario se han considerado 2000 conjuntos de tareas distintos, por tanto, en total se han generado 8000 conjuntos de tareas para la simulación experimental.

Respecto a la planificabilidad, tal y como puede verse en la Figura 6, los dos planificadores combinados muestran la mayor tasa de planificabilidad de todos los planificadores de tareas, seguidos de cerca por EDF0 y DM0, los cuales, como veremos a continuación, obtendrán malos resultados en los otros criterios.

Respecto a la interferencia producida, como se puede observar en la Figura 7, el P.C. especializado en minimizar la interferencia obtiene en promedio la menor interferencia de todas. Si bien es cierto que la diferencia en este caso entre todos los planificadores no es tan grande como lo puede ser en otros criterios.

Respecto a los cambios de contexto, como observamos en la Figura 8, el planificador combinado que minimiza los cambios de contexto obtiene en promedio la menor cantidad de cambios de contexto de todos los planificadores, seguido de cerca por los planificadores EDF 2 y DM 2, los cuales sacaban malos resultados en otros criterios, como la planificabilidad.

https://doi.org/10.17979/spudc.9788497498418.0870

	Parámetros experimentales			
Número de núcleos	2	4	6	8
Utilización Teórica	1.2	2.4	3.6	4.8
Número de tareas	8	16	24	32
Número de tareas con interferencia	3	6	9	12

Cuadro 1: Parámetros de la simulación realizada.

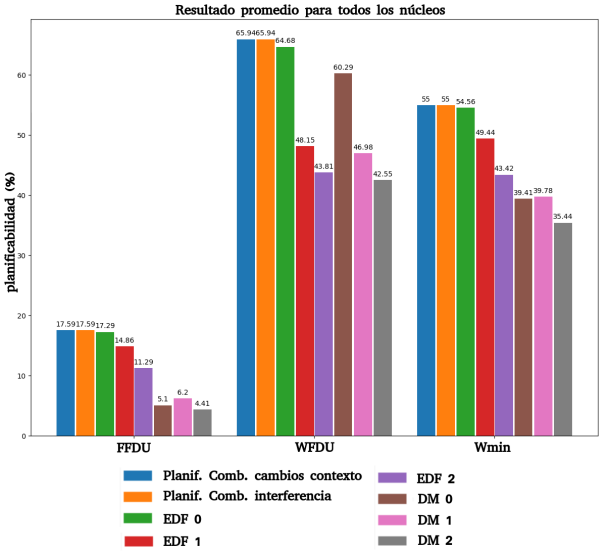

Figura 6: Comparativa de la planificabilidad por cada planificador

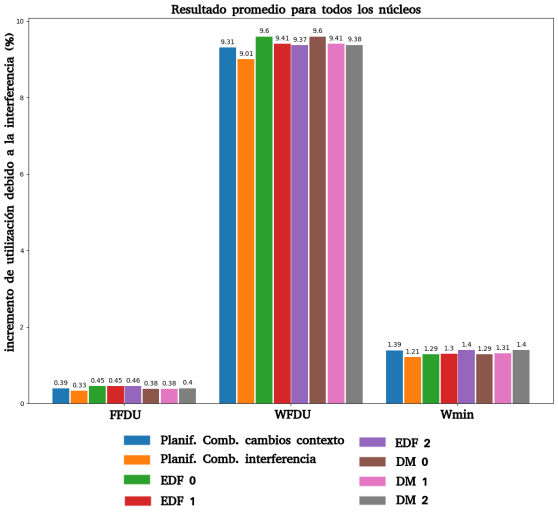

Figura 7: Comparativa de la interferencia producida en los planes por cada planificador

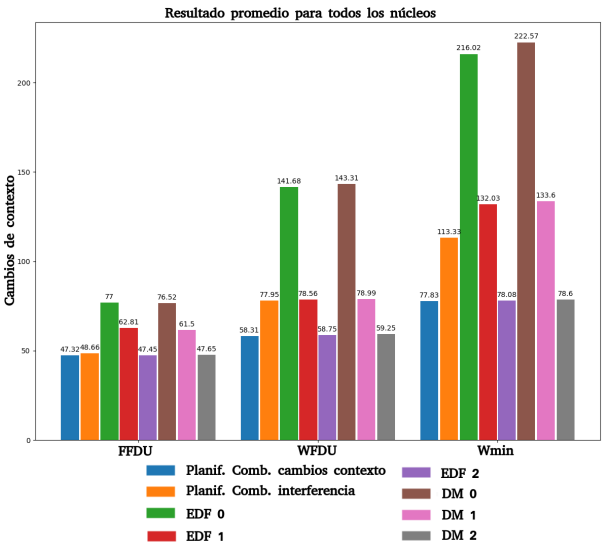

Figura 8: Comparativa del número de cambios de contexto promedio generados en los planes por cada planificador

7. Conclusiones

Como puede observarse en los resultados de la evaluación, los planificadores combinados obtienen los mejores resultados de entre todos los planificadores analizados. Lo cual es coherente dada la naturaleza del P.C., pues este usa en cada intervalo de ejecución la política que proporciona mejor resultado. Concluimos también que la flexibilidad de poder realizar una planificación con distinta política en cada intervalo es la clave para que el P.C. saque mejores resultados que la tradicional opción de mantener una política fija durante todo el hiperperiodo. Esto puede hacerse ya que en sistemas de tiempo real de alta criticidad, la planificación es estática.

Se puede concluir, por tanto, que el Planificador Combinado puede ser un planificador conveniente a utilizar en sistemas multinúcleo de tiempo real crítico donde la planificabilidad, los cambios de contexto o la interferencia son parámetros a optimizar.

Agradecimientos

Esta publicación es parte del proyecto de I+D+i PID2021-124502OB-C41, financiado por MCIN/

AEI/10.13039/501100011033.

English summary

Combined scheduler: a strategy for improving the performance of hard real-time systems

Abstract

This paper proposes and evaluates a new scheduling policy for real-time multicore systems. Combined scheduling tries to establish a new scheduling strategy that reduces context switches and interferences generated by the use of shared resources between cores. This strategy is based on the idea of choosing the best policy for a given execution interval throughout the plan. The result is a task scheduling implemented with different policies such as EDF (Earliest Deadline First) or DM (Deadline Monotonic) and variants of them. The proposal has been evaluated with different number of cores and load, obtaining improvements with respect to the use of a single scheduler over the entire execution horizon.

Keywords: real-time systems, scheduling, context switch, multicore.

Referencias

[1] Aceituno, J. M., Guasque, A., Balbastre, P., Simó, J., and Crespo, A. Hardware resources contention-aware scheduling of hard real-time multiprocessor systems. *Journal of Systems Architecture 118* (2021).

[2] Dasari, D., Andersson, B., Nelis, V., Petters, S. M., Easwaran, A., and Lee, J. Response time analysis of cots-based multicores considering the contention on the shared memory bus. In *2011IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications* (2011), pp. 1068–1075.

[3] Davis, R. I., and Burns, A. A survey of hard real-time scheduling for multiprocessor systems. *ACM Comput. Surv. 43*, 4 (Oct. 2011).

[4] Davis, R. I., Griffin, D., and Bate, I. Schedulability analysis for multi-core systems accounting for resource stress and sensitivity. In *33rd Euromicro Conference on Real-Time Systems, ECRTS 2021, July 5-9, 2021, Virtual Conference* (2021), vol. 196 of *LIPIcs*, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, pp. 7:1–7:26.

[5] Guasque, A., Aceituno, J. M., Balbastre, P., Simó, J., and Crespo, A. Schedulability analysis of dynamic priority real-time systems with contention. *The Journal of Supercomputing* (Apr. 2022).

[6] Guasque, A., Tohidi, H., Balbastre, P., Aceituno, J. M., Simó, J., and Crespo, A. Integer programming techniques for static scheduling of hard real-time systems. *IEEE Access 8* (2020), 170389–170403.

[7] Kim, H., de Niz, D., Andersson, B., Klein, M., Mutlu, O., and Rajkumar, R. Bounding memory interference delay in cots-based multi-core systems. In *2014 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)* (2014), pp. 145–154.

[8] Lampka, K., Giannopoulou, G., Pellizzoni, R., Wu, Z., and Stoimenov, N. A formal approach to the wcrt analysis of multicore systems with memory contention under phase-structured task sets. *Real-Time Systems 50* (11 2014), 736–773.

[9] Liu, C. L., and Layland, J. W. Scheduling algorithms for multiprogramming in a hard-real-time environment. *J. ACM 20*, 1 (Jan. 1973), 46–61.

[10] Lugo, T., Lozano, S., Fernández, J., and Carretero, J. A survey of techniques for reducing interference in real-time applications on multicore platforms. *IEEE Access 10* (2022), 21853–21882.

[11] Ripoll, I., Crespo, A., and Mok, A. K. Improvement in feasibility testing for real-time tasks. *Real-Time Syst. 11*, 1 (July 1996), 19–39.

 https://doi.org/10.17979/spudc.9788497498418.0870