Revista Iberoamericana de Automática e Informática Industrial

Revista Iberoamericana de Automática e Informática Industrial 21 (2024) 29-38

www.revista-riai.org

Técnicas de planificación para optimizar el rendimiento de los sistemas de tiempo real multiprocesador

José María Aceituno[a], Ana Guasque[a], Patricia Balbastre[a,*], José Simó[a], Carlos Eduardo Pereira[b], Alfons Crespo[a]

[a] *Instituto de Automática e Informática Industrial, Universitat Politècnica de València, Camino de Vera, s/n, 46022, Valencia, España.*
[b] *School of Engineering - UFRGS, Brasil*

To cite this article: Aceituno, J.M., Guasque, A., Balbastre, P., Simó, J., Pereira C.E., Crespo, A. 2024. Scheduling techniques for optimising the performance of multicore real-time systems. Revista Iberoamericana de Automática e Informática Industrial 21, 29-38. https://doi.org/10.4995/riai.2023.19935

Resumen

Los sistemas multinúcleo surgieron como alternativa y mejora a los tradicionales sistemas mononúcleo. Aunque el rendimiento de estos sistemas es mayor, poseen más complejidad. Además, su rendimiento puede verse degradado debido a que los recursos hardware compartidos introducen retrasos en la planificación. Para reducir este retraso o contención existen diversas técnicas, que se pueden aplicar tanto a la hora de alojar las tareas en los núcleos como al planificar las tareas dentro de cada núcleo. En este trabajo se propone un algoritmo de planificación que combina distintas políticas de planificación conocidas para obtener un plan temporal que posea una menor interferencia. Además, se propone una red neuronal artificial para predecir qué política de alojamiento se debe aplicar para minimizar la longitud de los intervalos que forman el plan temporal y así reducir la complejidad de planificación de cada intervalo.

Palabras clave: Sistemas de control de tiempo real, Planificación de sistemas de tiempo real, Sistemas ciber-físicos en control, Sistemas de control embebidos, Sistemas multiprocesador, Contención, Redes neuronales.

Scheduling techniques for optimising the performance of multicore real-time systems

Abstract

Multicore systems emerged as an alternative to traditional monocore systems. Although these systems possess high performance, they have more complexity. Moreover, their performance can be degraded because shared hardware resources introduce scheduling delays. To reduce this delay or contention, there are several techniques that can be applied both when allocating tasks to cores and when scheduling tasks within each core. In this paper, we propose a scheduling algorithm that combines different known scheduling policies to obtain a temporal plan that reduces the interference. In addition, we propose an artificial neural network to predict which allocation policy should be applied to minimize the length of the intervals that compose the temporal plan and thus reduce the scheduling complexity of each interval.

Keywords: Real-time control systems, Real-time scheduling, cyber physical systems, Embedded control systems, Multicore systems, Contention, Neural networks.

1. Introducción

Los sistemas multinúcleo se usan ampliamente dentro de sistemas empotrados de todo tipo, tanto en el sector industrial como para el uso de otras tecnologías más comunes. En muchos sistemas se requiere una exigencia alta de fiabilidad, como por ejemplo en la certificación de sistemas críticos. La solución en muchos casos pasa por una asignacion de recursos temporales de forma estática. Precisamente, en este artículo nos situaremos en el ámbito de los sistemas de planificación de tiempo real estáticos.

*Autor para correspondencia: patricia@ai2.upv.es

La realización de planificaciones para sistemas multinúcleo son problemas de tipo NP-complejo (*Nondeterministic Polynomial*). Para conseguir planes cíclicos que cumplan con los requisitos temporales que se exigen, es habitual recurrir a las técnicas heurísticas. Los problemas de los sistemas multinúcleo presentan cierta complejidad intrínseca por definición, además requieren de otras funcionalidades como la asignación o alojamiento (de tareas a núcleos), la gestión de la energía, la optimización del rendimiento, etcétera.

Además de los problemas mencionados anteriormente, en sistemas multinúcleo es más complejo calcular el tiempo de cómputo de las tareas debido al indeterminismo que introduce la compartición de recursos *hardware*.

Una de las fuentes de indeterminismo más recurrentes que aparecen en un sistema multinúcleo es la generación de interferencia. Esta se produce cuando tareas ejecutándose en diferentes núcleos intentan acceder simultáneamente a los mismos recursos (memoria, buses, ...). Este fenómeno puede retrasar la ejecución de las tareas e incluso podría comprometer la fiabilidad del sistema.

Uno de los grandes desafíos de las técnicas de planificación es evitar la interferencia. Para afrontarla, existen dos aproximaciones típicas, una genérica y otra específica: o bien utilizar un modelo específico que cuantifique la interferencia de un hardware concreto, o bien realizar propuestas generales que sean válidas para manejar la interferencia en cualquier tipo de hardware.

La primera aproximación nos proporciona una manera precisa para calcular la interferencia pero solo es válida para un tipo concreto de hardware, además esa interferencia calculada se añade al *worst case execution time* (WCET), y por tanto se obtiene un valor demasiado pesimista. La segunda aproximación obtiene un valor de interferencia mayor, pero al añadir ese valor al modelo temporal de manera independiente de su WCET, acabamos obteniendo un modelo menos pesimista. En este trabajo usamos un modelo temporal que incorpora el parámetro de la interferencia en línea con la segunda aproximación.

En esta área existen numerosos trabajos que afrontan la planificación de tareas y la planificabilidad en sistemas multinúcleos particionados de tiempo real. La planificación se realiza en dos fases, en la primera las tareas son asignadas a los núcleos, y en la segunda fase se planifican las tareas de cada núcleo.

Sin tener en cuenta el factor de la interferencia, la planificación de tareas en cada núcleo se realiza de forma independiente, pero si tenemos en cuenta el factor de la interferencia entonces la planificación de tareas en un núcleo afecta a la planificación de tareas en otro núcleo. Cabe señalar que en este segundo caso la gestión del tiempo es mucho más compleja.

La medición de los factores que crean la interferencia está fuera del alcance de este artículo, es decir, en este trabajo no consideramos el cálculo sobre cuánto tiempo se emplea en acceder a cada recurso compartido (por ejemplo, el acceso a una memoria RAM o a una caché).

El objetivo de este trabajo es proponer una nueva metodología de planificación para conseguir reducir el factor de interferencia. Para ello, por un lado proponemos un planificador que combina técnicas de planificación ya existentes y las combina para obtener una menor interferencia entre tareas en distintos núcleos. Dicho planificador genera unos intervalos de trabajo que serán clave para la siguiente fase. En la siguiente fase, introduciremos una técnica de *machine learning* (ML) para mejorar la eficiencia en la asignación de tareas a núcleos. El aprendizaje automático y, en particular, las redes neuronales del tipo Perceptron Multicapa son técnicas adecuadas para la síntesis de clasificadores o para aproximación de funciones cuando el problema presenta características no lineales. En este sentido se han aplicado con éxito en problemas de percepción sensorial y visión por computador. Concretamente proponemos una red neuronal que predice la conveniencia de usar una política de asignación sobre un conjunto de tareas concreto en cuanto al tamaño de los intervalos de planificación quie genera. Para esta parte, nos apoyaremos en una política de asignación de tareas a núcleos que reduce la interferencia. Esta política se llama Wmin y está propuesta en (Aceituno et al., 2021), al igual que el modelo temporal que vamos a utilizar.

En este trabajo nos situamos en el ámbito de los sistemas multinúcleos particionados de tiempo real sin migración de tareas.

Este artículo se organiza de la siguiente manera: en la sección 2 encontramos una lista de trabajos relacionados en esta área, tanto en planificación de tareas de sistemas multinúcleo, como también, otros trabajos donde se aplica ML a los sistemas de planificación de tareas. En la sección 3 encontramos una breve descripción del modelo temporal de tareas usado para la realización de este trabajo. En la sección 4 proponemos una nueva metodología de planificación de tareas para sistemas multinúcleos llamada planificador combinado. En la sección 5 se explica cómo afecta negativamente la longitud de los intervalos al sistema cuando son muy grandes. En esta sección también se presenta los beneficios que una política de asignación apoyada por ML puede ofrecer para solucionar este problema. En la sección 6 se explica detalladamente en qué consiste y cómo funciona la parte de ML propuesta en este trabajo. En la sección 7 se explica como se ha realizado la evaluación de la nueva metodología propuesta y cuáles han sido los resultados. Por último, en la sección 8 se determinan cuáles son las conclusiones de este trabajo.

2. Trabajos relacionados

Davis y Burns presentan en (Davis and Burns, 2011) una revisión de las metodologías de planificación en sistemas multinúcleo usadas desde 1960 hasta 2009. En dicho estudio se analizan los diferentes métodos y se presenta una taxonomía para la clasificación de las diversas técnicas.

La planificación de sistemas de tiempo real multiprocesador se realiza en dos fases: primero se alojan las tareas en núcleos (problema de asignación o alojamiento) y después se planifican las tareas en cada núcleo. Uno de los problemas que surgen a la hora de planificar tareas en núcleos es la aparición de la interferencia, que puede darse cuando dos tareas se ejecutan simultáneamente en diferentes núcleos.

Para el problema de asignación, existen diferentes heurísticas, como las analizadas en los trabajos de Oh et al.(Oh and Son, 1995) o Coffman et al.(Coffman et al., 1996), donde se proponen algoritmos de bin-packing como First Fit, Best Fit o Worst Fit. En el reciente trabajo de Aceituno et al. (Aceituno

et al., 2021), se propone un método de asignación de tareas a núcleos llamado Wmin que agrupa tareas que pueden provocarse interferencia mutuamente en un mismo núcleo.

Existen otros trabajos recientes que también estudian, recopilan y analizan el estado del arte respecto a los efectos de la interferencia en los sistemas de tiempo real particionados, como los trabajos de (Maiza et al., 2019) y (Lugo et al., 2022). En otros trabajos como el de Dasari et al., (Dasari et al., 2011) se estudian las fuentes de indeterminismo producidas por los recursos compartidos de hardware.

Para el problema de planificación, en (Altmeyer et al., 2015) se presenta un entorno de trabajo llamado Multicore Response Time Analysis (MRTA) con el que se puede medir y comparar el tiempo de respuesta para diferentes configuraciones de hardware de sistemas multinúcleo. Esas comparaciones son útiles dado que se pueden realizar durante la fase de diseño de un hardware. En este análisis no se utiliza el clásico concepto de WCET. Además, cabe señalar que en MRTA se omite la fase de asignación, es decir, se asume un alojamiento de tareas a núcleos ya dado.

Por otro lado, la aplicación de técnicas de inteligencia artificial (IA) en la planificación de sistemas a tiempo real no es todavía muy extensa, sin embargo existen algunos trabajos. Entre las diferentes técnicas de IA, el aprendizaje automático (*machine learning*, ML) se aplica de manera exitosa actualmente en áreas como la visión por computador, el procesamiento de lenguaje natural o la recomendación de sistemas. A pesar de que el ML está ganando fuerza en muchas áreas distintas, todavía no está muy implantado en el ámbito de la planificación de tareas en sistemas multinúcleo. No obstante, existen algunos trabajos desarrollados en esta área.

Respecto el empleo de redes neuronales en la planificación, en (Nemirovsky et al., 2017) se aplican a sistemas heterogéneos para obtener mejores rendimientos. Concretamente, aplica un predictor de redes neuronales para el rendimiento sobre *benchmarks* como SPEC2006 y SPLASH-2. El modelo de tareas usado aquí es diferente al de este artículo y se basa en sistemas heterogéneos. Sin embargo, nuestro trabajo se aplica a sistemas homogéneos.

En (B and Selvakumar, 2020) también se propone un asignador de carga de trabajo para sistemas multinúcleo, aunque se aplica en sistemas asimétricos. Este asignador está basado en un algoritmo de ML de tipo *deep learning* y busca mejorar el consumo de energía y el tiempo de ejecución.

También existen otros trabajos que aplican ML en la planificación de sistemas heterogéneos como (Shulga et al., 2016), en el cual aplican un algoritmo para mejorar el rendimiento del sistema. Concretamente, el algoritmo decide si usar la CPU o la GPU en función de la cantidad de datos de entrada. Este trabajo tiene cierta similitud al nuestro ya que ambos trabajos son en el área de la planificación estática y ambos predicen el recurso o método más apropiado en función a los datos de entrada. Aunque, entre otras diferencias, nuestro trabajo se enfoca en la predicción del algoritmo de alojamiento y no en el uso de CPU o GPU.

También encontramos otros trabajos importantes como (Zhang et al., 2017) donde se presenta un sistema de planificación basado en *clusters* que ayuda a reasignar durante la ejecución los recursos de un sistema. Esto lo hace basándose en predicciones sobre la carga de trabajo y demanda de recursos. Este sistema de *clusters* se llama SLAQ y se aplica para entrenamientos de ML.

3. Modelo temporal

Como se ha comentado anteriormente, este artículo está basado en el modelo de tareas para planificación estática de sistemas multinúcleo de tiempo real especificado en (Aceituno et al., 2021). Por tanto, en este trabajo tanto el planificador combinado como las políticas de asignación se implementan conforme a este modelo de tareas.

A continuación haremos un breve resumen de este modelo de tareas. Si se desea tener un mayor detalle del modelo, debe consultarse en (Aceituno et al., 2021).

Bajo este modelo, un sistema multinúcleo de m núcleos homogéneos ($M_0, M_1, M_2, ..., M_{m-1}$), puede alojar un conjunto de tareas τ con n tareas independientes. Cada tarea se representa con una tupla:

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

donde C_i es el peor caso de tiempo de ejecución, D_i es el plazo relativo, T_i es el periodo e I_i es la interferencia producida sobre otras tareas. En nuestro trabajo consideramos un modelo de plazos estrictos, es decir, los plazos nunca pueden ser mayores a los periodos: $D_i \leq T_i \quad \forall i$.

Como ejemplo, supongamos un sistema de 2 núcleos: M_0 y M_1, y 3 tareas τ_0, τ_1 y τ_2, con los siguientes parámetros: $\tau_0 = (1, 3, 3, 1)$, $\tau_1 = (1, 7, 7, 1)$, $\tau_2 = (1, 21, 21, 0)$. Con dichas tareas, podemos afirmar que τ_0 y τ_1 son tareas con interferencia dado que su valor de interferencia es positivo, es decir, $I_i > 0$. Supongamos que las tareas τ_0 y τ_2 se alojan en el núcleo M_0 y la tarea τ_1 se aloja en el núcleo M_1.

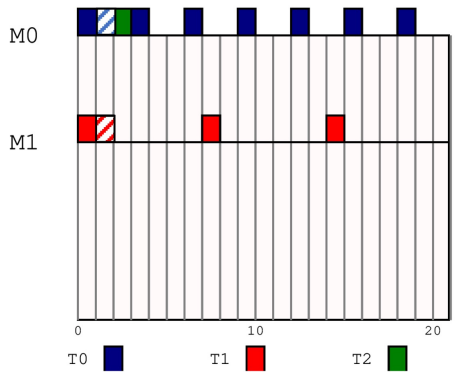

Figura 1: Ejemplo de cronograma bajo la política EDF

La novedad que introduce este modelo de tareas son los instantes de ejecución extra que se producen, y por tanto, el retraso en el tiempo de ejecución debido al factor de la interferencia. La Figura 1 muestra la planificación que obtenemos con estas tareas al aplicar la política *Earliest Deadline First* (EDF).

En la planificación podemos observar como en el primer instante de ejecución las tareas τ_0 y τ_1 coinciden, cada una en su núcleo. Dado que ambas tareas tienen interferencia ($I_0 = I_1 > 0$), en el siguiente instante se producirá 1 unidad de interferencia. Por tanto, en el segundo instante de ejecución del

cronograma podemos observar como cada tarea produce en la otra 1 instante de interferencia, es decir, la tarea τ_1 envía 1 unidad de interferencia al núcleo 0 y la tarea τ_0 envía 1 unidad de interferencia al núcleo 1. A nivel visual dichos instantes de interferencia los representamos como un bloque de líneas diagonales.

Por tanto, bajo este modelo de tareas, en este ejemplo la planificación en cada núcleo sufre un retraso de una unidad de tiempo debido a la interferencia.

3.1. Intervalos de trabajo

Una de las claves del trabajo que presentamos en este artículo es la reducción de la longitud de los intervalos de trabajo.

El concepto de intervalo de trabajo fue inicialmente definido para sistemas mononúcleo en (Lehoczky, 1990). El concepto de intervalo de trabajo puede definirse como:

Definición 1. *En sistemas multinúcleos particionados, un intervalo de trabajo (BP, Busy Period) es un intervalo de tiempo en el cual no hay ningún momento ocioso simultáneo en todos los núcleos del sistema.*

También puede entenderse como:

Definición 2. *Es un tramo de tiempo en el cual el procesador tiene designado en todo instante alguna tarea que ejecutar en algunos de sus núcleos.*

Como puede observarse en la Figura 2 La planificación de un conjunto de tareas en multinúcleo está formada por una sucesión de intervalos de trabajo. También podemos observar que el tamaño de los intervalos de trabajo pueden variar entre si.

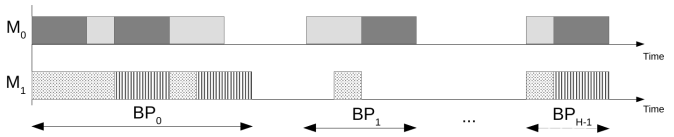

Figura 2: Ejemplo de Intervalos de trabajos (BP) durante una planificación

El concepto de intervalos de trabajo es clave en este artículo. En la siguiente sección proponemos un planificador de tareas que trabaja en base a los intervalos de trabajo y en otra sección posterior explicaremos una forma de reducir la longitud los intervalos de trabajo.

4. Planificador combinado

El planificador combinado (PC) es un algoritmo de planificación de tareas que combina distintas políticas de planificación para realizar un mismo plan. Dichas políticas pueden ser Earliest Deadline First (EDF) o Deadline Monotonic (DM) (Liu and Layland, 1973).

La peculiaridad del planificador combinado es que construye una planificación intervalo tras intervalo. En cada intervalo de trabajo el planificador combinado puede aplicar una política de planificación distinta. Para la elección de una política se comparan los resultados obtenidos de todas las políticas para el mismo intervalo. Es decir, cada intervalo se planifica con cada una de las políticas de planificación disponibles, después se compararán todos los resultados obtenidos y se escogerá la política que ofrezca el mejor resultado según el criterio deseado. En nuestro caso el criterio es la reducción de la interferencia.

El planificador combinado que presentamos en este artículo combina las políticas de planificación de EDF, DM y sus variantes que a continuación explicaremos. Cabe señalar que el planificador combinado podría ampliarse para que use y elija entre más políticas de planificación. Obviamente cuantas más políticas de planificación añadamos mayor será el tiempo de ejecución del PC. En cualquier caso, independientemente de las políticas de planificación que agreguemos al PC, la funcionalidad sería la misma. Lo único que podría cambiar son los resultados y el tiempo de ejecución.

Por tanto, tal y como dijimos anteriormente, bajo nuestra configuración del PC las políticas de planificación que tenemos son EDF, DM y las variantes de las mismas:

- Variante 1: Supongamos que una tarea se está ejecutando en el núcleo y otra tarea con más prioridad va a expulsar a la primera. En este caso, si a la tarea en ejecución le queda menos tiempo para acabar que el tiempo de ejecución de la otra tarea, entonces la tarea en ejecución hereda la prioridad de la otra tarea y por tanto no es expulsada.

- Variante 2: Consiste en no expulsar una tarea en ejecución durante N instantes de tiempo tanto si la tarea acaba de empezar su ejecución como si ha sido previamente expulsada.

Por tanto, al final el PC elegirá en cada intervalo entre 6 políticas de planificación: EDF original, EDF variante 1, EDF variante 2, DM original, DM variante 1 y DM variante 2.

A continuación vamos a ver con un ejemplo, como el PC prueba para cada intervalo de trabajo distintas políticas de planificación y elije aquella que menos interferencia genera.

4.1. Ejemplo de ejecución del PC

Asumimos un sistema de 2 núcleos: M_0 y M_1, con 4 tareas: τ_0, τ_1, τ_2 y τ_3 con los siguientes parámetros: $\tau_0 = (2, 6, 6, 0)$, $\tau_1 = (3, 10, 10, 1)$, $\tau_2 = (2, 7, 7, 1)$ y $\tau_3 = (3, 9, 9, 0)$.

Para simplificar el ejemplo, asumimos que el PC solo puede elegir entre 2 políticas de planificación: EDF0 (EDF original) y EDF1.

El PC comienza a planificar desde el instante 0 con el algoritmo EDF0 y lo hace hasta el instante 5, momento en el que el sistema se encuentra ocioso en todos sus núcleos, finalizando así ese primer intervalo. En esa planificación resultante, el PC se encuentra 0 instantes de interferencia. A continuación, el PC vuelve al instante inicial y planifica otra vez el mismo intervalo pero con otro algoritmo, en este caso EDF1. El resultado con EDF1 resulta ser idéntico al que se obtuvo anteriormente con EDF0, es decir, se obtienen 0 unidades de interferencia desde el instante 0 al 5. Ambas planificaciones de ese primer intervalo, con EDF0 y EDF1, pueden observarse en la Figura 3.

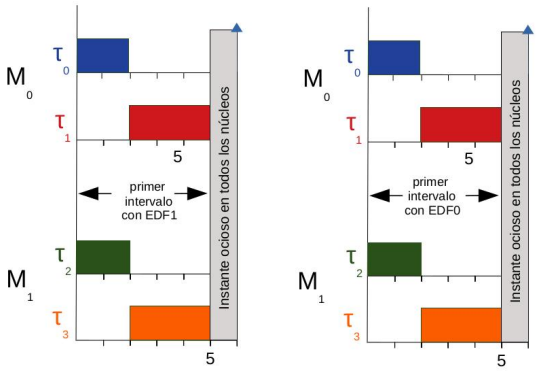

Figura 3: Cronograma resultante después de planificar el primer intervalo con EDF1 (a la izquierda) y con EDF0 (a la derecha).

Una vez se han probado todos las posibles algoritmos para el primer intervalo, el PC planificará con el que mejor resultados tenga, pero dado que en este caso tenemos un empate, el PC simplemente planificará el primer intervalo con el algoritmo que tenga asignado por defecto, en nuestro caso EDF0.

Una vez que el primer intervalo ya está resuelto, el algoritmo se sitúa en el instante 6, es decir, al inicio del segundo intervalo y realiza el mismo proceso otra vez. Tal y como puede observarse en la Figura 4, en este segundo intervalo, al planificar con el algoritmo EDF0 se obtendrían 2 unidades de interferencia. En la Figura 5 puede observarse que para ese mismo segundo intervalo, la planificación resultante con EDF1 obtendría 0 unidades de interferencia.

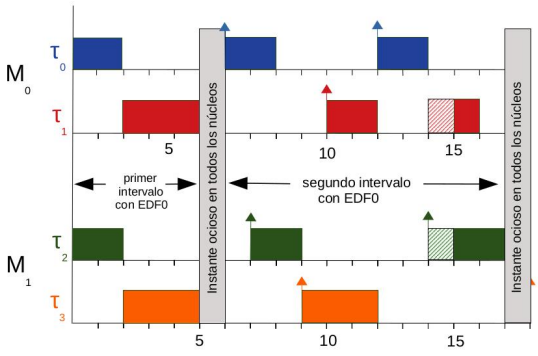

Figura 4: Cronograma resultante después de planificar el segundo intervalo con EDF0.

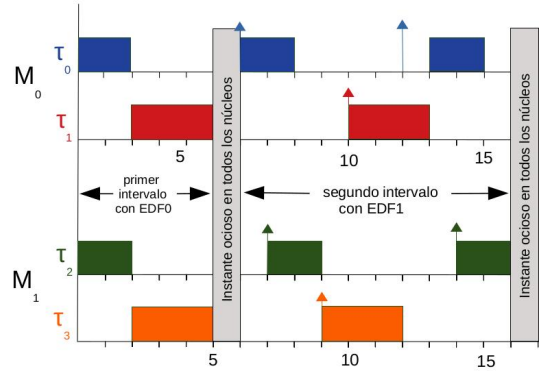

Figura 5: Cronograma resultante después de planificar el segundo intervalo con EDF1.

De ese modo, el PC elegiría EDF1 como el algoritmo que se aplique en el plan definitivo para el segundo intervalo. Hay que señalar que en este segundo intervalo, las duraciones de los intervalos resultantes son distintas, en el caso de EDF0 el intervalo comprende entre el instante 6 y el 17, sin embargo, en el caso de EDF1 el intervalo se extiende del instante 6 al 16, una unidad de tiempo menos. Esa diferencia de tamaño del intervalo está totalmente relacionada con la existencia, o no, de interferencia, si hay interferencia el intervalo será mayor.

El PC continuaría con la misma dinámica, intervalo tras intervalo, hasta llegar al final de la planificación, es decir, hasta el hiperperiodo.

Por tanto, el PC genera una planificación formada por diferentes intervalos, a los que llamaremos intervalos de trabajo (BP). En la siguiente sección se va a estudiar cómo afecta el modo en que se asignan las tareas a núcleos en la longitud de dichos intervalos de trabajo.

5. Efecto de la longitud de los intervalos de trabajo en la planificación

Como se ha visto en la sección anterior, en la fase de planificación se divide el espacio total temporal (hiperperido) en los intervalos de trabajo. El PC o el planificador elegido actúa en cada intervalo. Cuanto más pequeños sean los intervalos, más rápido se obtendrá una solución. Si en lugar del PC o cualquier otra heurística, utilizáramos algún algoritmo más costoso computacionalmente, podríamos tener problemas para encontrar una solución en un tiempo razonable.

En esta sección, se analiza qué efecto tiene el algoritmo de asignación de tareas a núcleos en la longitud de los intervalos.

Para el estudio de este comportamiento hemos analizado dos algoritmos de asignación, uno que presenta muy buenos resultados de planificabilidad y otro que obtiene muy buenos resultados de interferencia: Worst Fit Decreasing Utilisation (WFDU) y Wmin.

- Worst Fit Decreasing Utilisation (WFDU) (Oh and Son, 1995) está basado en la heurística Worst Fit (WF), en la que cada tarea se aloja en el núcleo más vacío siempre que sea posible y, si no, se abre un nuevo núcleo. DU indica que las tareas se ordenan por utilización decreciente.
- Wmin (Aceituno et al., 2021) es un asignador de tareas a núcleos que tiene en cuenta la interferencia a la hora de alojar las tareas. Intenta, siempre que sea posible, agrupar las tareas que provocan/reciben interferencia en el mismo núcleo.

Para comprobar cómo afecta el algoritmo de alojamiento usado a la longitud de los intervalos, hemos realizado una simulación de 1000 conjuntos de tareas, con 8 tareas por conjunto, asignadas a un sistema homogéneo de 2 núcleos. Por cada conjunto de tareas se han aplicado dos técnicas de alojamiento (WFDU y WMin). Posteriormente, los conjuntos de tareas han sido planificados por el PC, obteniendo en cada caso las longitudes de los correspondientes BPs.

De ese modo, el PC a partir de 1000 conjuntos de tareas ha realizado un total 2000 planificaciones (1000 basadas en el alojamiento de WFDU y 1000 en el de Wmin).

De todos los intervalos de los que se compone cada planificación, se escoge el intervalo de mayor longitud para evaluar la influencia de los algoritmos de alojamiento. En la Figura 6 se representa el tamaño del intervalo más grande para cada planificación resultante.

Tal y como puede observarse en la Figura 6, el resultado de esta experimentación es que la política de asignación de Wmin (cruces rojas), en la mayoría de los casos, provoca tamaños de intervalos de trabajo máximos más grandes que con WFDU (cruces azules). Esto es una desventaja para Wmin respecto a WFDU.

Por otro lado, sabemos que Wmin tiene la ventaja de producir planificaciones con menos interferencia que WFDU tal y como se ha evaluado en (Aceituno et al., 2021).

Sin embargo, en la Figura 6 también se puede observar que no en todos los casos Wmin produce BP grandes. Es decir, para algunos conjuntos de tareas, Wmin es el asignador ideal en todos los criterios porque nos genera una planificación con una baja interferencia y además los BP generados no son grandes.

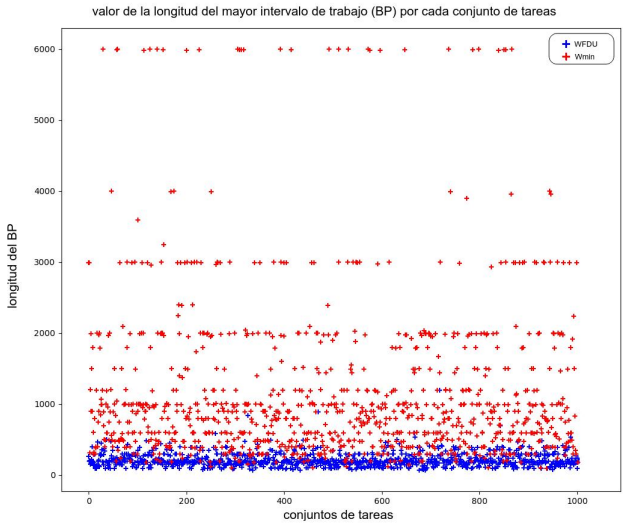

Figura 6: Longitud máxima de los BP en cada planificación por cada conjunto de tareas.

Por tanto, en esta sección se ha demostrado que la longitud de los intervalos de trabajo se ve afectada por el algoritmo que aloja las tareas en los núcleos. ¿Por qué es importante la longitud de estos intervalos? Porque cuanto mayor sea esta longitud, más costoso será calcular la planificación de cada intervalo. En el caso de que la planificación la realice el PC, este coste puede no ser muy significativo pero si aplicamos técnicas no convencionales como programación lineal entera u otro tipo de algoritmos computacionalmente más complejos, puede resultar no factible la planificación en intervalos grandes. En la siguiente sección se va a estudiar si, aplicando técnicas de IA, se puede seleccionar el algoritmo de alojamiento que provoca intervalos de trabajo más cortos para cada conjunto de tareas. Con esta información, se aplicaría el PC a aquellos intervalos marcados como grandes y otras técnicas de optimización más costosas en tiempo para los intervalos pequeños.

Para ello, se va a proponer una red neuronal artificial para predecir si, en un conjunto de tareas dado, el alojador Wmin proporciona intervalos de trabajo pequeños o grandes.

6. Predictor de intervalos basado en red neuronal artificial

Tal y como se menciona al final de la sección anterior, en este artículo proponemos un modelo de red neuronal artificial que, a partir de un conjunto de tareas dado, sea capaz de predecir como será la longitud de los BP generados (grandes o pequeños), en caso de que dicho conjunto de tareas sea alojado en los núcleos por el algoritmo Wmin y planificado por el PC.

Para el desarrollo de la red neuronal, se han probado múltiples configuraciones, utilizando el conjunto de herramientas de código abierto Numpy, Keras y Tensorflow.

A continuación presentamos las características de la red neuronal que mejor resultados nos ha proporcionado y que es la usada para la obtención de los resultados finales mostrados en la sección de evaluación:

- Como algoritmo de la red, se ha elegido Adam (Adaptive moment estimation). Adam es un típico algoritmo usado para optimización estocástica de redes neuronales y está basado en el descenso de gradiente (P. Kingma, 2015).

- La función de coste usada es una función de entropía cruzada binaria, que es típicamente utilizada para la función de coste en machine learning y optimización. En el lenguaje usado, pytorch, dicha función corresponde a BCEloss().

- La función de activación: la función de activación usada en cada neurona es la sigmoide, la cual siempre devolverá un valor entre 0 y 1. La Función Sigmoide es idónea para problemas de ajuste de funciones no lineales.

- Número de neuronas en cada capa: Las capas de entrada y salida tienen un número de neuronas determinado por la naturaleza del problema planteado, es decir, la capa de entrada es un bloque de 24 números enteros que corresponden a las 3 características de cada una de las 8 tareas. Por cada tarea tenemos: tiempo de cómputo, periodo y la interferencia que produce la tarea. La capa de salida está compuesta por 1 sola neurona que devuelve un valor entre 0 y 1. Idóneamente dicho valor será 1 o próximo a 1 cuando la red prediga que el conjunto de tareas suministrado en la entrada no generará BP grandes.

- Capas ocultas: la red contiene 2 capas ocultas de 50 neuronas cada una. Durante la investigación del modelo se probaron distintas cantidades de capas (desde 1 hasta 16) con variables cantidades de neuronas por cada capa (desde 24 hasta 512).

- Origen de los datos de entrenamiento: Para la creación de los datos de entrenamiento (y test) hemos utilizado el mismo generador de tareas usado en (Aceituno et al., 2021), A dicho generador de tareas se le ordenó crear 3000 conjuntos de tareas. En la sección de Evaluación se explica con más detalle el formato de dichos conjuntos de tareas.

- Épocas: Tras probarse distintos valores para el número de épocas (de 100 a 2000), los mejores resultados se obtuvieron con entrenamientos de 400 épocas.

- Normalización: la red presentó mejores resultados cuando los datos de entrada introducidos estaban normalizados.

En la Figura 7 podemos observar la estructura general de la red neuronal utilizada en este trabajo.

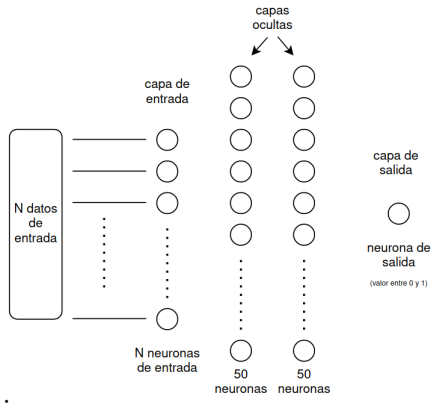

Figura 7: Estructura de la red neuronal.

7. Evaluación

A continuación presentamos la sección de evaluación de PC y del predictor de intervalos.

7.1. Resultados sobre el PC

En esta sección se comentan los resultados del planificador combinado y la metodología usada para evaluarlo. Para un análisis más completo y también para comprobar la versatilidad del PC, hemos implementado 2 versiones del PC. En cada versión se optimiza un criterio distinto. La primera versión del PC reduce la interferencia tal y como se explica en la sección del PC. La otra variante se ha implementado con el criterio de los cambios de contexto, es decir, el PC reduce al mínimo el número de cambios de contexto en la planificación.

Para evaluar el PC se han considerado 4 escenarios distintos. Cada escenario considera una cantidad diferente de núcleos: 2, 4, 6 y 8 núcleos. Para cada escenario hemos generado de manera aleatoria 2000 conjuntos de tareas distintos. Por tanto, en esta simulación experimental empleamos un total de 8000 conjuntos de tareas. Los detalles de los 4 escenarios están descritos en el Cuadro 1.

Para evaluar los resultados del PC, lo comparamos con otros algoritmos planificadores típicos. Dos de los planificadores más comunes en la planificación de tareas son EDF, del cual ya hemos hablado anteriormente y Deadline Monotomic (DM), otro algoritmo planificador de tareas típicamente usado en sistemas de tiempo real. Además, para una comparación más amplia, aplicamos tanto a EDF como a DM las 2 variantes explicadas en la sección del PC. De ese modo, obtenemos un total de 6 planificadores distintos: EDF original (EDF0), EDF1, EDF2, DM original (DM0), DM1 y DM2. Por tanto comparamos los resultados de estos 6 planificadores con los resultados de las 2 versiones del PC.

Como se puede observar en el cuadro 1, para cada cantidad de núcleos usamos conjuntos de tareas con distinta configuración. Los parámetros de los conjuntos de tareas son: la utilización teórica, el número de tareas y el número de tareas con interferencia. Aunque los valores de los parámetros para cada núcleo son distintos, se mantiene siempre una proporcionalidad. Por ejemplo, si para 2 núcleos la utilización teórica es 1.2, entonces para cuatro núcleos la autorización teórica es el doble, es decir, 2.4. El hiperperiodo para todos los conjuntos de tareas está limitado como máximo a 3000 unidades de tiempo.

Respecto a la fase de asignación, hemos utilizado los dos métodos comentados anteriormente, Wmin y WFDU. Es decir, cada conjunto de tareas es alojado por dos métodos distintos y planificado por 8 planificadores diferentes. Esto ocurre en los 4 escenarios.

Para la evaluación de los distintos planificadores, utilizamos 3 criterios: la planificabilidad de los conjuntos de tareas, la interferencia producida en las planificaciones y los cambios de contexto que se producen en cada planificación.

A continuación comentamos los resultados obtenidos en función a cada uno de los criterios.

Respecto a la planificabilidad, podemos observar en la Figura 8 que el PC es el planificador que saca mayor porcentaje de planificabilidad en sus 2 versiones. Esto ocurre tanto en WFDU como en Wmin aunque en ambos casos EDF original se queda bastante próximo a los resultados del PC.

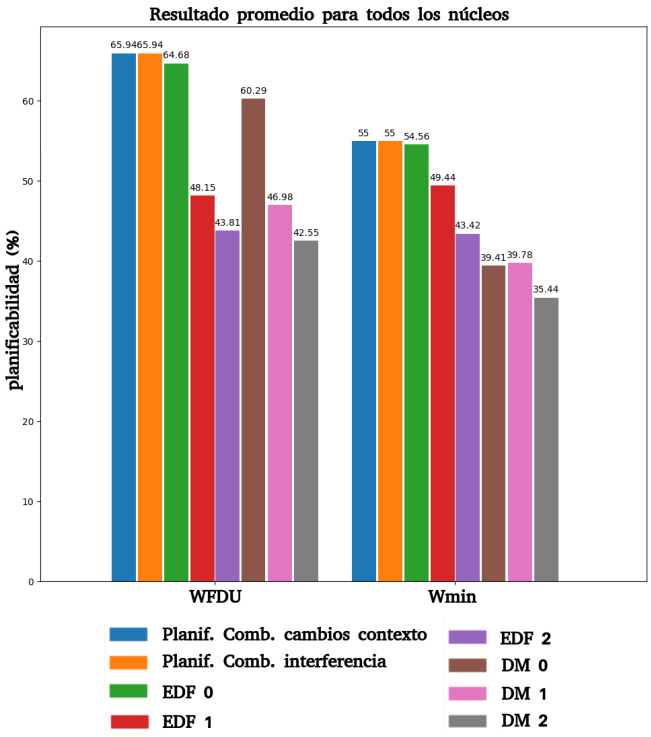

Figura 8: Comparativa de la planificabilidad por cada planificador.

Respecto a la interferencia, podemos observar en la Figura 9 que la versión del PC que reduce la interferencia es el planificador que obtiene mejor resultado. Especialmente en el caso de WFDU el PC saca mayor diferencia al resto de planificadores.

Tabla 1: Parámetros de la simulación realizada.

	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

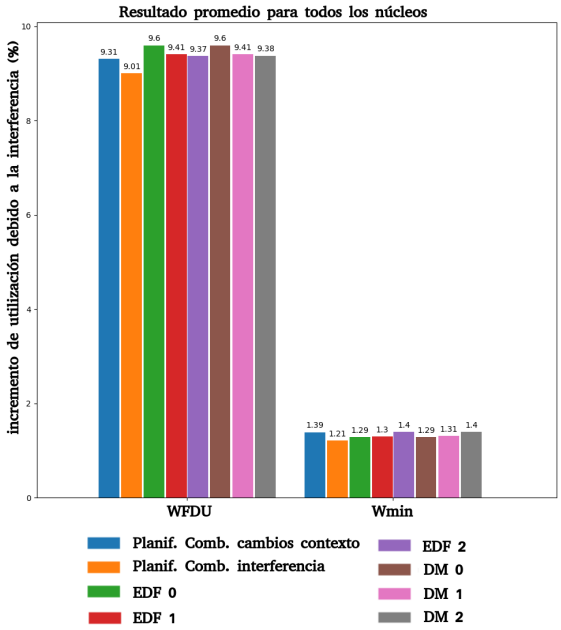

Figura 9: Comparativa de la interferencia en los planes por cada planificador.

Respecto a los cambios de contexto, podemos observar en la Figura 10 que los resultados entre los planificadores son muy dispares. En esta gráfica se muestra como el PC en su versión de cambios de contexto obtiene claramente el mejor resultado. Esto ocurre tanto en WFDU como en Wmin.

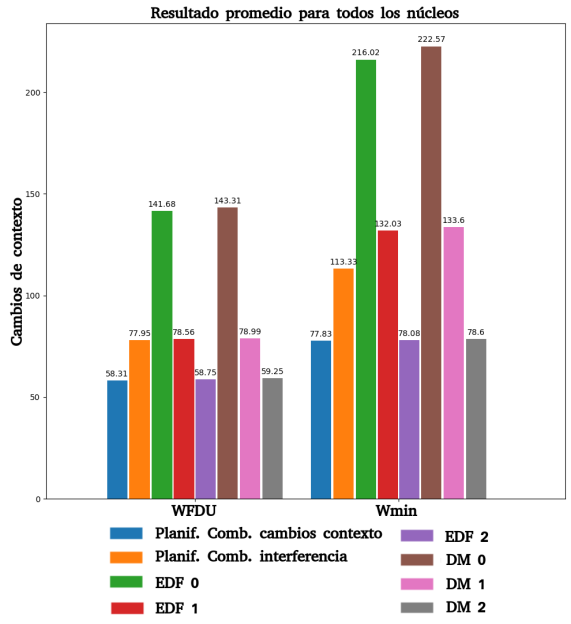

Figura 10: Comparativa del número de cambios de contexto promedio.

7.2. *Resultados sobre la red neuronal*

Para entrenar la red neuronal, hemos generado 2550 conjuntos de tareas con las siguientes características:

- 8 tareas por cada conjunto.
- 3 de dichas tareas con interferencia.
- la interferencia generada es el 15 % del valor del WCET de la tarea (valor teórico dado que se redondea obligatoriamente porque los instantes de tiempo son siempre enteros).
- Periodos iguales a plazos.
- Utilización teórica del 75 %, por tanto 1,5 respecto a 2 núcleos.
- Y un hiperperiodo máximo de 6000 instantes de tiempo.

Los 3000 conjuntos de tareas han sido alojados por la política de asignación Wmin. Tras alojar las tareas, se planifican todos los conjuntos de tareas con el PC, y tras dichas planificaciones se obtienen los intervalos (BP).

A partir de este punto, nuestro objetivo es probar nuestro modelo de red neuronal para que prediga si un conjunto de tareas que es alojado por Wmin y planificado por el PC acabará generando algún intervalo grande en su planificación (consideramos un intervalo grande como mayor a 100 instantes de tiempo).

Para ello, se han generado 450 conjunto de tareas nuevos con características similares a los utilizados para entrenar la red neuronal.

En la Figura 11 podemos observar el resultado final de la red neuronal usada. En ella se muestran una serie de marcas o cruces de 2 colores, azul y rojo, distribuidas por una gráfica. Las cruces azules representan aquellos conjuntos de 8 tareas que al ser alojados en los núcleos con la política Wmin ocurre que, tras planificar con el PC, todos los intervalos generados son pequeños, es decir, son menores a 100 instantes de tiempo. Por contra, las marcas rojas representan lo contrario, aquellos conjuntos de tareas que tras planificar con el Planificador Combinado, entre los intervalos generados habrá al menos alguno mayor a 100 instantes de tiempo.

Como puede observarse la mayoría de las marcas azules están en la parte superior de la gráfica, cercanas al valor 1 y la mayoría de las marcas rojas están en la parte inferior de la gráfica, cercanas al 0. Esto significa que la red neuronal tiene una relativa buena precisión de predicción respecto a los nuevos conjuntos de tareas que se le pasan a la red dado que a simple vista en la Figura 11 vemos muy pocas marcas azules próximas al 0 y pocas marcas rojas próximas al 1.

Si redondeamos los valores de la salida, a 0 o 1, obtenemos una precisión de la red neuronal de 85.64 %. Es decir, la probabilidad evaluada en el test de la red neuronal nos indica que hay un 85.64 % de posibilidades de que la red prediga correctamente si un conjunto de tareas genere o no genere tras la planificación con el PC un intervalo mayor a 100 unidades de tiempo.

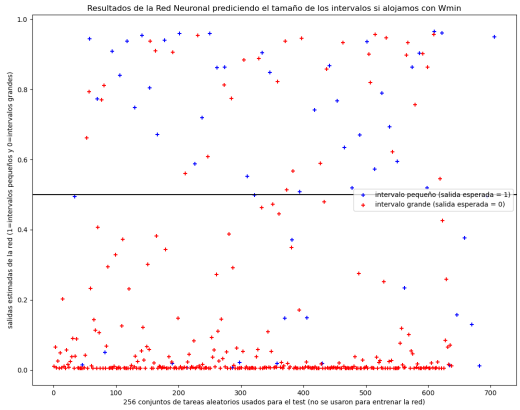

Figura 11: Salidas de la red neuronal.

Si queremos analizar el resultado de manera más precisa, podemos observar la Figura 12 donde se muestran solo los conjuntos de tareas que de antemano sabemos que generan BP mayores a 100 unidades de tiempo y por tanto se espera que la salida de la red sea más cercana al valor 0 que al 1. Como podemos observar en dicha gráfica, la red neuronal predice en la mayoría de casos con un valor inferior a 0.5, es decir, acierta con su predicción en la mayoría (91.8 %). Por tanto podemos afirmar que la red neuronal detecta muy bien cuando un conjunto de tareas generará BP grandes si es alojado con Wmin.

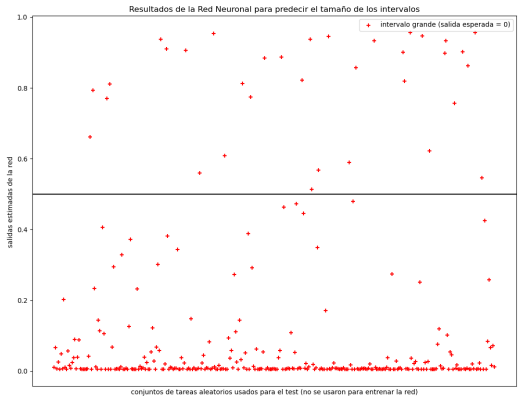

Figura 12: Predicciones de la red neuronal para intervalos grandes.

Por contra, también podemos analizar el resultado de solo los conjuntos de tareas que de antemano sabemos que generan BP menores a 100 unidades de tiempo y por tanto se espera que la salida de la red sea más cercana al valor 1 que al 0. Dicha gráfica se representa en la Figura 13 y observamos que la red acierta en la mayoría de casos (64.9 %), aunque con menor tasa que en el caso opuesto.

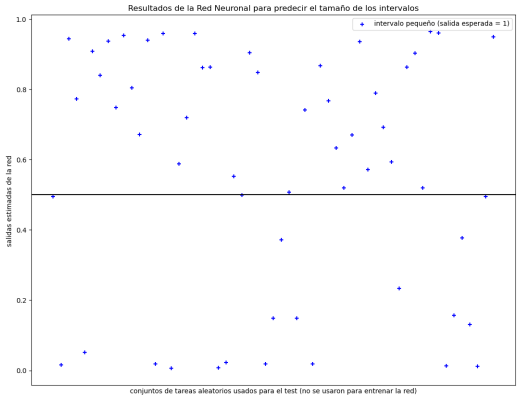

Figura 13: Predicciones de la red neuronal para intervalos pequeños.

8. Conclusiones

Tras los resultados obtenidos en la evaluación, podemos concluir que los 2 aportes para la planificación de tareas que introducimos en este trabajo presentan resultados que demuestran una mejora en la eficiencia de la planificación.

En el caso del PC, seleccionando el criterio de decisión más adecuado, demostramos que obtiene mejores resultados en los 3 criterios evaluados (planificabilidad, interferencia y cambios de contexto). Si el PC escoge entre todas las políticas de planificación aquella que provoca menor interferencia, entonces el PC obtiene el mejor resultado en cuanto a reducción de interferencia. Si el PC escoge entre todas las políticas de planificación aquella que provoca menores cambios de contexto, la diferencia con EDF es bastante notable. Esto es bastante significativo dado que a pesar que EDF obtenía una planificabilidad próxima al PC, sin embargo se comprueba que EDF obtiene peores resultados que el PC en cambios de contexto.

Por otro lado, observamos que los modelos predictivos de Machine Learning pueden ayudar a mejorar el rendimiento de los sistemas de planificación de tiempo real en planificación estática para sistemas multinúcleo. En este caso, no se obtiene una reducción de interferencia directamente sino que se evalúa la bondad de un asignador de tareas a núcleos que genere intervalos de trabajo pequeños. Esto es conveniente para utilizar técnicas de planificación basadas en optimización de la interferencia. En este trabajo aplicamos la predicción para el algoritmo Wmin, pero en futuros trabajos se plantea aplicarlo a otros algoritmos de asignación. De ese modo, se abre la posibilidad de que se pueda elegir un algoritmo de asignación para un conjunto de tareas en función al resultado de las predicciones.

Es importante indicar que el modelo de red neuronal propuesto se aplica a conjuntos de 8 tareas de ciertas características. Si se deseara aplicar un modelo similar en otro sistema que maneje otro tipo de conjuntos de tareas, debería configurarse y entrenarse otro modelo de red con otros parámetros.

Dicho de otro modo, este trabajo no propone un modelo de red neuronal genérico y válido para cualquier tipo de carga y sistema, pero sí demuestra que es posible utilizar modelos de red en esta línea de investigación.

Por tanto se deja la puerta abierta a la extensión de este trabajo tanto para la búsqueda de modelos de ML que se puedan aplicar a sistemas con conjuntos de tareas de otras características, como también la búsqueda de modelos de ML más genéricos aplicables a sistemas diversos. Con respecto al PC, en trabajos futuros se pretende implementar un criterio multiobjetivo para la toma de decisión de la mejor política, de forma que no se minimicen o bien los cambios de contexto o bien la interferencia, si no que puedan combinarse varios objetivos en el criterio de decisión.

Agradecimientos

Esta publicación es parte del proyecto de I+D+i PLEC2021-007609 financiado por MCIN/ AEI/ 10.13039/501100011033 y por "Unión Europea NextGenerationEU / PRTR" y del proyecto de I+D+i PID2021-124502OB-C41, financiado por MCIN/ AEI/10.13039/501100011033. También ha sido financiado por PAID-10-20 (Universitat Politècnica de València).

Referencias

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

Altmeyer, S., Davis, R. I., Indrusiak, L., Maiza, C., Nelis, V., Reineke, J., 2015. A generic and compositional framework for multicore response time analysis. In: Proceedings of the 23rd International Conference on Real Time and Networks Systems. RTNS '15. p. 129–138.

B, G., Selvakumar, J., 2020. Appropriate allocation of workloads on performance asymmetric multicore architectures via deep learning algorithms. Microprocessors and Microsystems 73, 102996.
DOI: https://doi.org/10.1016/j.micpro.2020.102996

Coffman, E. G., Garey, M. R., Johnson, D. S., 1996. Approximation Algorithms for Bin Packing: A Survey. PWS Publishing Co., USA, p. 46–93.

Dasari, D., Andersson, B., Nelis, V., Petters, S. M., Easwaran, A., Lee, J., 2011. 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. pp. 1068–1075.

Davis, R. I., Burns, A., oct 2011. A survey of hard real-time scheduling for multiprocessor systems. ACM Comput. Surv. 43 (4).

Lehoczky, J., 1990. Fixed priority scheduling of periodic task sets with arbitrary deadlines. In: [1990] Proceedings 11th Real-Time Systems Symposium. pp. 201–209.
DOI: 10.1109/REAL.1990.128748

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

Lugo, T., Lozano, S., Fernández, J., Carretero, J., 2022. A survey of techniques for reducing interference in real-time applications on multicore platforms. IEEE Access 10, 21853–21882.
DOI: 10.1109/ACCESS.2022.3151891

Maiza, C., Rihani, H., Rivas, J. M., Goossens, J., Altmeyer, S., Davis, R. I., jun 2019. A survey of timing verification techniques for multi-core real-time systems. ACM Comput. Surv. 52 (3).

Nemirovsky, D., Arkose, T., Markovic, N., Nemirovsky, M., Unsal, O., Cristal, A., 2017. A machine learning approach for performance prediction and scheduling on heterogeneous cpus. In: 2017 29th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD). pp. 121–128.
DOI: 10.1109/SBAC-PAD.2017.23

Oh, Y., Son, S. H., Nov. 1995. Allocating fixed-priority periodic tasks on multiprocessor systems. Real-Time Syst. 9 (3), 207–239.
DOI: 10.1007/BF01088806

P. Kingma, D., 2015. Adam: A method for stochastic optimization. conference ICLR 2015.

Shulga, D. A., Kapustin, A. A., Kozlov, A. A., Kozyrev, A. A., Rovnyagin, M. M., 2016. The scheduling based on machine learning for heterogeneous cpu/gpu systems. In: 2016 IEEE NW Russia Young Researchers in Electrical and Electronic Engineering Conference (EIConRusNW). pp. 345–348.
DOI: 10.1109/EIConRusNW.2016.7448189

Zhang, H., Stafman, L., Or, A., Freedman, M. J., 2017. Slaq: Quality-driven scheduling for distributed machine learning. In: Proceedings of the 2017 Symposium on Cloud Computing. SoCC '17. Association for Computing Machinery, p. 390–404.
DOI: 10.1145/3127479.3127490