Balanceo Dinámico con Control de Consumo en Sistemas Multinúcleo de Tiempo Real

José Luis March, Julio Sahuquillo, Salvador Petit, Houcine Hassan y José Duato [1]

Resumen— **El consumo de energía es un asunto de diseño crítico en los sistemas empotrados actuales. Muchos sistemas aplican técnicas de *Dynamic Voltage Scaling (DVS)*, que permiten cambiar dinámicamente la velocidad del sistema en función de la características de la carga. En este contexto, para manejar eficientemente la energía, la carga debe estar adecuadamente equilibrada entre los núcleos.**

Este artículo propone una nueva heurística para balancear la carga en un sistema empotrado con un procesador multinúcleo multihilo de grano grueso. Esta heurística está ideada para mejorar el tiempo de solapamiento entre memoria y procesador, manteniendo la utilización de los núcleos equilibrada. Con este fin, la heurística controla dinámicamente el nivel de frecuencia/voltaje para garantizar el cumplimiento de las tareas de tiempo real estricto, así como para conseguir una buena relación entre deadlines perdidos y ahorro energético para las tareas de tiempo real flexible. Los resultados experimentales muestran que con la heurística propuesta, el consumo normalizado presenta valores entre un 8 % y un 70 % dependiendo de la agresividad del planificador.

Palabras clave— **Balanceo, Planificación, Consumo, Tiempo Real, Sistemas Empotrados, Multinúcleo**

I. Introducción

DEBIDO a los avances tecnológicos el consumo energético se ha convertido en un aspecto de diseño crucial en los microprocesadores modernos. Este hecho es incluso más crítico en sistemas de tiempo real (e.g., robótica, redes de sensores) y sistemas empotrados (e.g., PDAs, smartphones), donde la duración de la batería es vital [1]. Por eso, cómo reducir el consumo de energía se ha convertido en un tema de investigación primordial en estos sistemas [2], [3].

Dynamic Voltage Scaling (DVS) [2] es una técnica que permite al sistema trabajar a diferentes niveles de frecuencia/voltaje. En función de si cada núcleo tiene su propio regulador DVS o si se utiliza el mismo regulador para proporcionar voltaje a todos los núcleos (obligándolos a trabajar a la misma velocidad), DVS se puede clasificar en *local* y *global*, respectivamente. *DVS Local* es más complejo y caro porque se precisan más reguladores de frecuencia/voltaje. De esta forma, muchos procesadores multinúcleo usan un único dominio de reloj ya que usar dominios de reloj independientes es todavía mucho más caro [1]. Además, está demostrado que si la carga se equilibra entre los núcleos de forma apropiada, *DVS global* puede ser tan eficiente como *DVS local* [4]. Por eso hay varios procesadores de altas prestaciones, como el IBM Power 7 [5], que implementan *DVS global*.

El problema del consumo se agrava aún más [6] con el creciente número de procesadores en cada chip, que es la tendencia actual. En esos procesadores, donde varias unidades de cómputo están disponibles, los sistemas de tiempo real incluyen un distribuidor de carga que reparte las tareas entre esas unidades de acuerdo a una heurística dada [7], [8]. Típicamente, estas heurísticas han tratado sólo el sistema de distribución; sin embargo, debido al problema del consumo recientemente se han propuesto heurísticas para ahorrar energía. En este contexto, varios trabajos han analizado el consumo revisando heurísticas previas como Worst Fit o Best Fit [9], [10]. Respecto a técnicas multihilo, este trabajo se centra en un procesador multinúcleo multihilo de grano grueso debido a la buena relación que ofrece entre consumo energético y planificabilidad en tiempo real [11].

Este artículo presenta una heurística de distribución dirigida a incrementar el tiempo de solapamiento entre accesos a memoria y tiempo de cómputo. Este algoritmo distribuye las tareas entre los núcleos balanceando la demanda de memoria hasta que se alcanza un umbral de utilización dado. Luego, como todos los núcleos trabajan a la misma velocidad (DVS global), se sigue la política del núcleo menos cargado para balancear los requerimientos de velocidad. A diferencia de entornos de tiempo real estricto (*Hard Real-Time o HRT*) donde el cumplimiento de los deadlines debe estar garantizado, en entornos de tiempo real flexible (*Soft Real-Time o SRT*) se permite la pérdida de deadlines. En este contexto, cuanto más alta es la frecuencia del sistema, menos deadlines se pierden, y viceversa. El algoritmo propuesto varía la velocidad del sistema y usa un umbral como indicador para la pérdida de deadlines aceptable. Con este fin, se han ideado diversos modos de agresividad para aumentar y disminuir la frecuencia. Las técnicas propuestas se han evaluado sobre un modelo de un microprocesador empotrado ARM, usado en sistemas comerciales (i.e., HTC Hero, iPhone), ejecutando una serie de benchmarks standard.

El resto del artículo está estructurado de la siguiente forma. La Sección II discute el trabajo relacionado. En la Sección III se describen el sistema modelado, la heurística propuesta y el planificador. La Sección IV muestra los resultados experimentales. Finalmente, la Sección V presenta unas breves conclusiones.

[1]Departamento de Informática de Sistemas y Computadores (DISCA), Universitat Politècnica de València (UPV), e-mail: jomarcab@gap.upv.es, {jsahuqui,spetit,husein,jduato}@disca.upv.es

II. Trabajo Relacionado

En multiprocesadores, donde la planificación energéticamente eficiente de tareas de tiempo real es un problema NP-Hard [9], [1], existen dos categorías dependiendo de si la cola de tareas es compartida por todos los procesadores (*global scheduling*) o no (*partitioned scheduling*). En el primer caso, las tareas pueden migrar entre procesadores y se seleccionan las más prioritarias para que se ejecuten. Por ejemplo, en [12] Kato et al. presentan un planificador basado en EDF (*Earliest Deadline First*) global para tareas esporádicas. En el último caso, cada tarea es asignada a un procesador de forma permanente y no puede migrar. Usando esta técnica el problema de la planificación en multiprocesadores se divide en varios problemas de planificación en uniprocesadores. Por lo tanto, se suelen adoptar algoritmos conocidos de la teoría de los uniprocesadores como EDF y RMS (*Rate Monotonic Scheduling*). Por ejemplo, en [10] AlEnawy et al. consideran el problema de la minimización de energía para tareas HRT periódicas preferentes planificadas en un multiprocesador simétrico (*Symmetric Multiprocessor o SMP*) con DVS.

Respecto a sistemas multihilo, en [13] Park et al. proporcionan una gestión de energía garantizando las prestaciones para aplicaciones multihilo en procesadores multinúcleo. En [14] Cazorla et al. presentan una arquitectura donde un multihilo simultáneo interactúa con el sistema operativo para mejorar la predictibilidad en tiempo real. Sin embargo, ninguno de estos trabajos considera planificación en tiempo real y gestión de energía en conjunto. En [15] El-Haj-Mahmoud et al. afirman que los procesadores multihilo simultáneos dificultan la prueba de las garantías HRT, y proponen virtualizarlos en múltiples procesadores superescalares monohilo. No obstante, este trabajo no trata el ahorro energético.

En lo referente a las heurísticas de distribución, recientemente se han publicado muchas propuestas. En [9] Aydin et al. desarrollan un entorno donde se usa el balanceo de carga para producir ditribuciones de tareas de tiempo real energéticamente eficientes en un sistema SMP con DVS. Estas distribuciones se evalúan usando el algoritmo EDF. En [10] los autores se centran en el impacto de las heurísticas en la viabilidad y el consumo, y proponen una nueva que reserva un subconjunto de procesadores para tareas ligeras. Finalmente, Seo et al. [1] presentan una heurística que migra tareas entre los núcleos para mantener las utilizaciones de los mismos balanceadas. A diferencia de este trabajo, todos estos artículos no aprovechan el solapamiento de memoria-procesador entre diferentes hilos para reducir el consumo.

III. Modelo del Sistema

El sistema modelado, mostrado en la Figura 1, esta compuesto por una fuente y un sumidero de tareas HRT y SRT, un distribuidor de carga, un planificador con control de consumo, y un procesador multinúcleo con m núcleos. Cada núcleo es un procesador multihilo de grano grueso que modela un ARM11 MPCore, implementando tres hilos hardware. Los planificadores comparten un regulador DVS global que ajusta la velocidad y el consumo de todos los núcleos del sistema a los requerimientos de sus planificadores EDF.

El sistema ejecuta tareas periódicas tanto de tiempo real estricto como flexible. Una tarea se puede lanzar a ejecución al principio de cada periodo activo y, en el caso de las tareas HRT, terminará obligatoriamente su ejecución antes de alcanzar su deadline. Para un proceso de planificación más manejable se asume que el final del periodo y el deadline de una tarea coinciden. También hay algunos periodos en los que las tareas no se ejecutan por no estar activas (i.e., periodos inactivos). Esta secuencia consecutiva de periodos activos e inactivos permite modelar cambios de modo como en sistemas reales.

A parte del periodo y el deadline, una tarea también se caracteriza por su WCET (*Worst Case Execution Time*). La utilización de una tarea es usada por varios planificadores y distribuidores para comprobar si la planificabilidad del conjunto de tareas es viable o no, y se obtiene de la siguiente forma: $U = \frac{WCET}{Periodo}$

A. Heurísticas de Distribución

Los procesadores multihilo de grano grueso cambian el hilo en ejecución cuando ocurre un evento de larga latencia (e.g., un acceso a memoria). Esto permite solapar la ejecución de diversas tareas, siempre que una de ellas esté consumiendo CPU mientras que el resto están accediendo a memoria o esperando para ello. En un sistema de tiempo real, la consecuencia de este solapamiento es que las tareas menos prioritarias pueden ejecutar instrucciones en el procesador o acceder a memoria. Por tanto, pueden terminar antes que si se ejecutaran en un procesador con un solo hilo. Esta propiedad permite al sistema ahorrar tiempo extra que puede ser usado para reducir la velocidad y el consumo. Nótese que este tiempo extra puede aumentar cuando las tareas son complementarias respecto a los requisitos de memoria y procesador. Por consiguiente, el objetivo

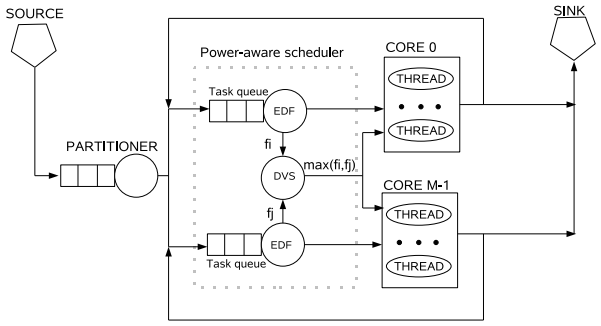


Fig. 1. Sistema Modelado.

de la heurística propuesta es incrementar el tiempo de solapamiento de memoria y procesador. Con este fin, la heurística distribuye tareas complementarias entre los núcleos disponibles.

Sin embargo, distribuir las tareas de esta forma independientemente de la utilización de cada núcleo podría llevar a distribuciones de carga poco equilibradas. En esta situación, sería imposible reducir globalmente el consumo bajando la velocidad, porque la utilización acumulada de un núcleo fuerza a ejecutar su carga a altas velocidades para garantizar los deadlines de sus tareas HRT. Por lo tanto, la heurística debe proveer un buen balanceo de memoria entre los núcleos para incrementar el solapamiento, pero también limitando la utilización de cada núcleo.

La política propuesta, denominada *Load-Bounded Memory Balancing (LBMB)*, está resumida en la Figura 2. El algoritmo distribuye tareas entre los núcleos asignando la tarea con más consumo de memoria al núcleo con menos consumo acumulado de memoria. Luego, actualiza el conjunto de tareas a ser distribuidas, así como el consumo de memoria acumulado y la utilización del núcleo destino.

Este comportamiento se mantiene siempre que la utilización acumulada del núcleo destino tras la asignación no supere un umbral dado, el cual se obtiene considerando la utilización de los núcleos asumiendo que la carga está perfectamente equilibrada. Si este umbral se supera, entonces la tarea es asignada al núcleo menos cargado. En este sentido, el algoritmo persigue balancear la carga para reducir tanto velocidad como consumo.

B. Planificador HRT con Control de Consumo

El planificador con control de consumo está compuesto por un planificador EDF por cada núcleo y el controlador DVS global. Cada planificador EDF tiene dos funciones. Primero, el planificador selecciona la velocidad mínima a la cual las tareas de este núcleo son planificables siguiendo el algoritmo EDF [16] e informa de este requisito al controlador DVS global. En segundo lugar, como cada núcleo puede ejecutar tres hilos hardware, lanza a ejecución hasta tres tareas, aquellas más prioritarias dentro del conjunto de tareas. Respecto a EDF, estas tareas son aquellas cuyo deadline expira antes. Si algún hilo está ocupado por una tarea menos prioritaria, ésta cede su lugar a la más prioritaria.

1: **Algorithm:** Load-Bounded Memory Balancing (LBMB)
2: **Input:** $Tasks$: task set to be distributed
3: **Output:** $Tasks_1, Tasks_2, ..., Tasks_m$: tasks sets assigned to the different cores
4: Let $U_{th} = \frac{\sum_{i=1}^{n} U_i}{m}$
5: **while** $Tasks$ is not empty **do**
6: $target_task \leftarrow max : \forall task \in Tasks, MEM_{max} \geq MEM_{task}$
7: $target_core \leftarrow min : \forall core, MEM_{min} \leq MEM_{core}$
8: **if** $U_{target_core} + U_{target_task} > U_{th}$ **then**
9: $target_core \leftarrow min : \forall core, U_{min} \leq U_{core}$
10: **end if**
11: $U_{target_core} \leftarrow U_{target_core} + U_{target_task}$
12: $MEM_{target_core} \leftarrow MEM_{target_core} + MEM_{target_task}$
13: $Tasks_{target_core} \leftarrow Tasks_{target_core} \cup \{target_task\}$
14: $Tasks \leftarrow Tasks - \{target_task\}$
15: **end while**

Fig. 2. Heurística Load-Bounded Memory Balancing.

TABLA I
ENERGÍA (E) USADA POR FRECUENCIA (F).

F[MHz]	500	400	300	200	100
E[pJ/ciclo]	450	349.2	261.5	186.3	123.8

Nótese que estas acciones deben realizarse cada vez que una tarea llega o sale del sistema debido a dos razones. La primera es porque cuando el conjunto de tareas cambia puede requerirse una velocidad distinta, y la segunda razón es que puede suceder que debido al buen solapamiento de las tareas que se ejecutan en un mismo núcleo, cuando una tarea termina su ejecución, hay más tiempo extra (i.e., el resto de tareas terminará antes), permitiendo al planificador EDF cumplir las restricciones de tiempo real a velocidades más bajas.

Una vez que la lógica de control del DVS global recibe los requisitos de velocidad de todos los planificadores EDF del sistema, se proporciona a los núcleos un nivel adecuado de frecuencia/voltaje para cumplir ésos requisitos. Los cinco niveles de frecuencia/voltaje considerados en este trabajo se han elegido teniendo en cuenta los del procesador Pentium M [17] mostrados en la Tabla I.

C. Soportando Tareas SRT

Cuando se ejecuta una carga compuesta sólo por tareas HRT, se establecerá una frecuencia más alta de la estrictamente necesaria debido al tiempo de solapamiento memoria-procesador transcurrido. No obstante, establecer una frecuencia menor podría suponer la pérdida de deadlines, lo cual no está permitido en sistemas HRT. Por contra, considerando tareas SRT, utilizar una frecuencia menor que la requerida por los planificadores EDF puede conllevar importantes beneficios de energía. Esto puede hacerse ya que las tareas SRT toleran cierto porcentaje de deadlines perdidos.

Cuando se combinan tareas HRT y SRT en la misma carga, la frecuencia del sistema está limitada por la frecuencia requerida por la planificabilidad de las tareas HRT. Además, las tareas SRT tendrán menor prioridad que las HRT. Por lo tanto, las tareas SRT sólo se ejecutarán cuando no haya ninguna tarea HRT requiriendo el procesador.

Para dar soporte a este comportamiento, el planificador consta de dos colas de tareas distintas,

```
FREQUENCY_CALCULATION()
if workload_change
  then
        FREQ_hard ← hard_calculation()
        switch
          case mode_1 :
                FREQ ← FREQ_hard − 100
          case mode_2 :
                FREQ ← FREQ_hard − 200
          case mode_3 :
                FREQ ← FREQ_hard − 300
          case mode_4 :
                FREQ ← FREQ_hard − 400
if activations = w
  then
        activations ← 0
        if lost_deadlines > th
          then FREQ ← FREQ_current + 100
        lost_deadlines ← 0
```

Fig. 3. Cálculo de Frecuencia.

una para las tareas HRT y otra para las SRT. Para obtener la frecuencia del sistema, el planificador calcula la frecuencia requerida por la carga entera (tareas tanto HRT como SRT) con la que no se perderían deadlines. Luego, la frecuencia se reduce en función del modo de ahorro de energía seleccionado. Cuatro modos diferentes se han implementado con diferentes niveles de agresividad. Estos modos se denominan modo-1, modo-2, modo-3 y modo-4, donde el número indica cuántos niveles de frecuencia se reducen respecto a la frecuencia requerida por el planificador EDF. Por ejemplo, si la frecuencia requerida es 400 MHz entonces el modo-1 y el modo-2 reducirían esta frecuencia a 300 MHz y 200 MHz, respectivamente. La Figura 3 muestra el cálculo de frecuencia.

En este sistema la frecuencia debe ser comprobada (y cambiada si se requiere) en dos casos. El primer caso corresponde al caso descrito para tareas HRT. Además, la frecuencia debe ser comprobada también cuando se pierde un deadline de una tarea SRT. En este caso, la frecuencia se subirá un nivel cuando el número de deadlines supere un umbral th (*threshold*). El algoritmo comprueba el número de deadlines perdidos para una tarea dada tras w ejecuciones de esa tarea, lo cual se denomina tamaño de ventana (*window size*). Los contadores de deadlines perdidos para cada tarea se incrementan cada vez que se pierde un deadline y se reinician al final de la ventana.

En la Figura 4 se observa el comportamiento de los diferentes modos de comprobar el umbral bajo un mismo escenario. Por ejemplo, un tamaño de ventana de $w = 4$ incrementa la frecuencia al final de la segunda ventana, porque se supera el umbral de pérdidas admisibles (i.e., 2), mientras que una ventana de $w = 12$ no modifica la frecuencia, ya que el número de deadlines perdidos es sólo 5 en la primera ventana y 2 en la segunda.

IV. Resultados Experimentales

El sistema ha sido evaluado sobre Multi2Sim [18], un entorno de simulación de procesadores multihilo multinúcleo detallado ciclo a ciclo. Se ha asumido ejecución en orden, ancho de etapa de dos instrucciones y una latencia de memoria de 100 ciclos. Multi2Sim ha sido ampliado para implementar el distribuidor, el planificador con control de consumo tanto para tareas HRT como SRT y el modelo de frecuencias. Más detalles acerca de las ampliaciones requeridas para modelar este tipo de sistemas pueden encontrarse en [19]. Nótese que aunque el algoritmo se puede aplicar a cualquier número de núcleos, este trabajo se centra en dos y cuatro núcleos.

Para los experimentos se han usado un conjunto de benchmarks del repositorio Real-Time Systems Benchmark [20]. Con estos benchmarks se han diseñado diferentes mezclas para explorar los beneficios en ahorro de energía. Estas mezclas se han diseñado teniendo en cuenta aspectos como la utilización de las tareas, su periodicidad, y la secuencia de periodos activos e inactivos. Por simplicidad se ha asumido que las tareas HRT no sólo son las más prioritarias sino que también su planificabilidad en el modo de ejecución más agresivo está asegurada. El objetivo de estas mezclas es comprobar cómo las prestaciones de las tareas SRT se alteran por la presencia de las tareas HRT en el sistema. Las mezclas se han diseñado para simular cómo se comporta el sistema en condiciones de sobrecarga. En ese sentido, la velocidad requerida por el procesador será una de las más altas del sistema, así que el algoritmo debe compensar los deadlines perdidos con ahorro de energía.

A. Reducción de Consumo y Deadlines Perdidos

Para obtener la energía consumida con una mezcla, ésta se ejecuta hasta su hiperperiodo. Luego, se obtiene el tiempo que el procesador trabaja a cada nivel de frecuencia/voltaje. Por último, la energía acumulada se calcula multiplicando dicho tiempo por su correspondiente energía consumida por ciclo.

Nótese que es importante reducir el tiempo que se trabaja a frecuencias altas ya que el consumo normalizado para una tecnología dada (e.g., 45 nm) crece exponencialmente con la frecuencia del reloj [1]. Así, el planificador debe concentrarse en los niveles de frecuencia/voltaje más eficientes.

Teniendo en cuenta tanto las tareas HRT como las

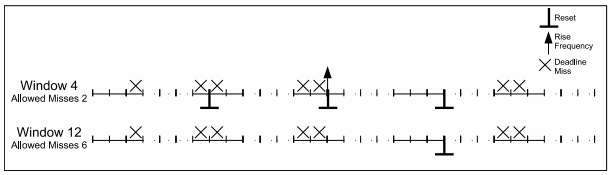
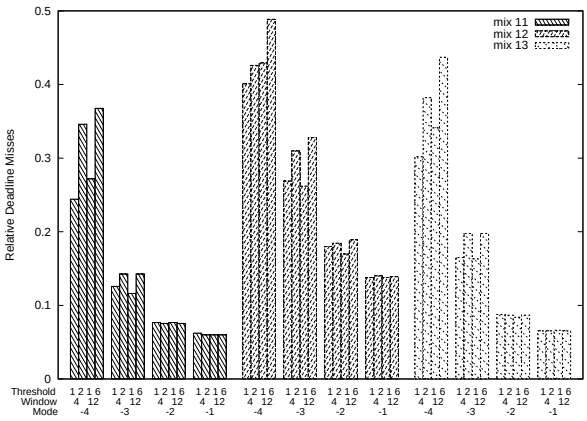
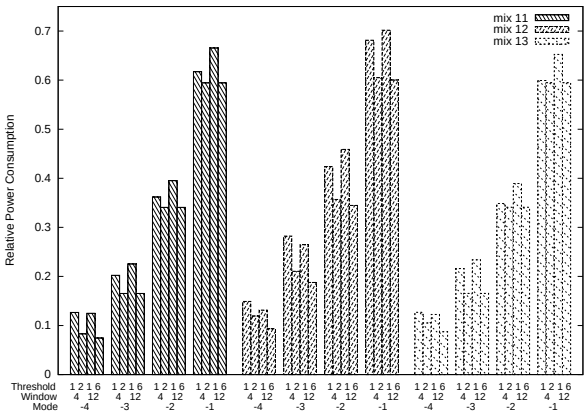


Fig. 4. Tamaños de Ventana.

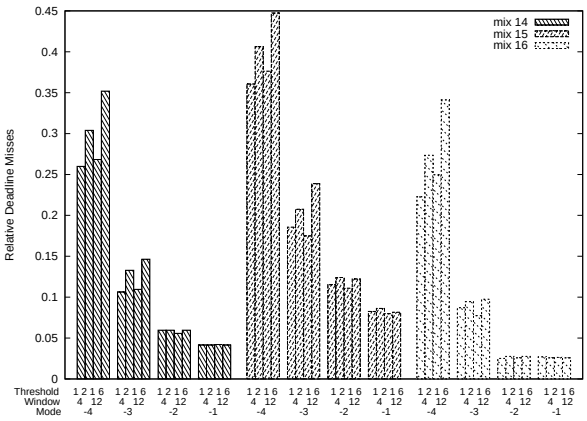


(a) Deadlines Perdidos.

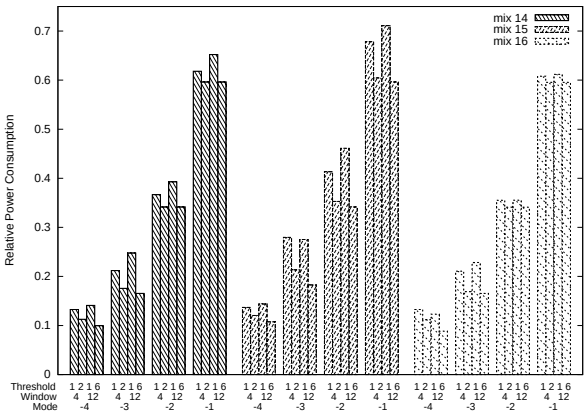


(b) Energía.

Fig. 5. Energía y Deadlines Perdidos (2 Núcleos).



(a) Deadlines Perdidos.



(b) Energía.

Fig. 6. Energía y Deadlines Perdidos (4 Núcleos).

SRT, se debe evaluar no sólo el consumo sino también la calidad de servicio (QoS) en términos de deadlines perdidos. En los experimentos se ha variado el tamaño de ventana ($w = 2$ and $w = 4$), el umbral ($th = 1$ y $th = 2$ para $w = 4$, y $th = 1$ y $th = 6$ para $w = 12$) y el modo de ejecución (-1, -2, -3 y -4). Las Figuras 5 y 6 muestran los deadlines perdidos y el consumo normalizados, para la heurística LBMB aplicada en un sistema compuesto por tareas HRT y SRT con 2 y 4 núcleos, respectivamente. La energía se ha normalizado respecto al consumo total del sistema trabajando siempre a la máxima frecuencia. El porcentaje de deadlines perdidos se ha obtenido al dividir el número de deadlines perdidos por el total de ejecuciones de tareas SRT.

Los resultados muestran que los gráficos de deadlines perdidos y consumo tienen formas opuestas, es decir, cuanta más energía se consume, menos deadlines se pierden. Esto se debe a que cuando se aumenta la frecuencia, hay más tiempo extra disponible y por eso las tareas SRT se planifican fácilmente, aunque se consume más energía.

Respecto a los tres parámetros mostrado en las Figuras 5 y 6 (umbral, tamaño de ventana y modo) hay que decir que el modo es el que tiene un mayor impacto tanto en consumo de energía como deadlines perdidos. Si el objetivo principal del sistema es minimizar los deadlines perdidos la mejor opción sería el modo-1, mientras que si la prioridad del sistema es reducir el consumo una buena elección sería el modo-4. Para requisitos intermedios el modo-2 y el modo-3 serían más adecuados.

Nótese que los modos menos agresivos (-1 y -2) pierden menos deadlines, pero los modos más arriesgados consumen menos energía. Como se ha comentado, el sistema tiene que encontrar una solución de compromiso entre éstos parámetros para establecer la frecuencia. En ese sentido, el sistema usa el umbral para determinar el riesgo que se asume en términos de deadlines perdidos. Un umbral alto puede reducir el consumo pero a costa de permitir más deadlines perdidos. La influencia del umbral es más notoria en los dos modos más agresivos (-4 y -3), como se aprecia en la Figura 5(a). Esto se debe a que en los modos menos arriesgados (-1 y -2) trabajar a altas frecuencias previene al sistema de perder un excesivo número de deadlines, ya que el umbral no es alcanzado tantas veces como en los modos arriesgados. También puede verse que dado un umbral par el modo-4, un tamaño de ventana mayor implica una reducción de consumo pero también un aumento en los deadlines perdidos.

A pesar de las consideraciones anteriores, los tamaños de ventana usados obtienen importantes reducciones en el consumo, y este parámetro debería

ser ajustado en función de las características de la carga, esto es, considerando el número de repeticiones de las tareas. Por ejemplo, en un sistema en el que la mayoría de los benchmarks se ejecutan sólo de 5 a 8 veces durante el hiperperiodo, una ventana de 4 sería más apropiada y conseguiría mejores resultados que una ventana de 12. Una vez se elige un modo, el sistema puede priorizar los requerimientos de *QoS* de diferentes formas durante su tiempo de vida. Por ejemplo, el sistema podría empezar permitiendo pocas pérdidas de deadlines, pero al final de la misión del sistema, cuando quede poca batería, puede ser más útil permitir la pérdida de más deadlines para reducir el consumo y aprovechar al máximo lo que quede de batería.

Respecto a la escalabilidad, las mezclas lanzadas en el procesador con cuatro núcleos (14, 15 y 16) se han diseñado siguiendo el mismo criterio que las mezclas lanzadas en el procesador con dos núcleos (11, 12 y 13). Comparando la Figura 5 y la Figura 6, hay que decir que el planificador muestra un buen comportamiento al aumentar el número de núcleos tanto en número de deadlines perdidos como en consumo de energía.

V. Conclusiones

Este artículo presenta un distribuidor y un planificador de tareas HRT y SRT para un procesador multinúcleo de grano grueso implementado en una amplia gama de sistemas empotrados. El planificador garantiza los deadlines de las tareas HRT y reduce energía aplicando técnicas de DVS. Respecto al distribuidor, una nueva heurística para repartir la carga se ha presentado. La heurística propuesta distribuye las tareas entre los núcleos en función de sus requisitos de memoria, con tal de incrementar el tiempo de solapamiento y, por tanto, conseguir tiempo extra que puede ser dedicado a reducir el consumo de energía.

Los resultados muestran que cuando se aplica la heurística propuesta (LBMB) en un sistema con tareas tanto HRT como SRT, el consumo relativo de energía varía entre un 8 % y un 70 %, y los deadlines perdidos entre un 3 % y un 48 %, dependiendo de los deadlines que se permitan perder y de la frecuencia seleccionada. Por lo tanto, la *QoS* requerida es la clave para determinar tanto el consumo energético como los deadlines perdidos. Entonces, permitir un alto número de deadlines perdidos de tareas SRT puede ayudar a reducir mucho el consumo a costa de aumentar efectivamente los deadlines perdidos, y viceversa.

Agradecimientos

Este trabajo ha sido financiado por el Ministerio de Ciencia e Innovación (MICINN), el programa Consolider y los fondos del Plan E, así como por los fondos FEDER de la Comisión Europea, mediante las subvenciones a los proyectos CSD2006-00046 y TIN2009-14475-C04-01.

Referencias

[1] E. Seo, J. Jeong, S. Park, and J. Lee, "Energy Efficient Scheduling of Real-Time Tasks on Multicore Processors," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 11, pp. 1540–1552, 2008.

[2] C. Hung, J. Chen, and T. Kuo, "Energy-Efficient Real-Time Task Scheduling for a DVS System with a Non-DVS Processing Element," in *Proceedings of the 27th RTSS*, Rio de Janeiro, Brazil, 5-8 December 2006, pp. 303–312.

[3] R. Ubal, J. Sahuquillo, S. Petit, H. Hassan, and P. López, "Power Reduction in Advanced Embedded IPC Processors," *Intelligent Automation and Soft Computing*, vol. 15, no. 3, pp. 495–507, 2009.

[4] J. Donald and M. Martonosi, "Techniques for Multicore Thermal Management: Classification and New Exploration," in *ISCA'33*, Boston, USA, 2006, pp. 78–88.

[5] R. Kalla, B. Sinharoy, W. J. Starke, and M. Floyd, "Power7: IBM's Next-Generation Server Processor," *IEEE Micro*, vol. 30, no. 2, pp. 7–15, 2010.

[6] G. R. Goud, N. Sharma, K. Ramamritham, and S. Malewar, "Efficient Real-Time Support for Automotive Applications: A Case Study," in *Proceedings of the 12th RTCSA*, Sydney, Australia, 2006, pp. 335–341.

[7] J. López, J. Díaz, M. García, and D. García, "Worst-Case Utilization Bound for EDF Scheduling on Real-Time Multiprocessor Systems," in *Proceedings of the 12th ECRTS*, Stockholm, Sweden, June 2000, pp. 25–33.

[8] A. Burchard, J. Liebeherr, Y. Oh, and S. Son, "New strategies for Assigning Real-Time Tasks to Multiprocessor Systems," *IEEE Transactions on Computers*, vol. 44, no. 12, pp. 1429–1442, 1995.

[9] H. Aydin and Q. Yang, "Energy-Aware Partitioning for Multiprocessor Real-Time Systems," in *Proceedings of the 17th IPDPS, Workshop on Parallel and Distributed Real-Time Systems*, Nice, France, April 2003, p. 113.

[10] T. A. AlEnawy and H. Aydin, "Energy-Aware Task Allocation for Rate Monotonic Scheduling," in *Proceedings of the 11th RTAS*, San Francisco, USA, 2005, pp. 213–223.

[11] A. El-Haj-Mahmoud and E. Rotenberg, "Safely Exploiting Multithreaded Processors to Tolerate Memory Latency in Real-Time Systems," in *Proceedings of the CASES*, Washington, DC, USA, 2004, pp. 2–13.

[12] S. Kato and N. Yamasaki, "Global EDF-based Scheduling with Efficient Priority Promotion," in *Proceedings of the 14th RTCSA*, Taiwan, 2008, pp. 197–206.

[13] S. Park, W. Jiang, Y. Zhou, and S. Adve, "Managing Energy-Performance Tradeoffs for Multithreaded Applications on Multiprocessor Architectures," in *the SIGMETRICS*, San Diego, USA, 2007, pp. 169–180.

[14] F. Cazorla, P. Knijnenburg, R. Sakellariou, E. Fernández, A. Ramirez, and M. Valero, "Predictable Performance in SMT Processors: Synergy between the OS and SMTs," *IEEE Transactions on Computers*, vol. 55, no. 7, pp. 785–799, 2006.

[15] A. El-Haj-Mahmoud, A.AL-Zawawi, A. Anantaraman, and E. Rotenberg, "Virtual Multiprocessor: An Analyzable, High-Performance Architecture for Real-Time Computing," in *the CASES*, USA, 2005, pp. 213–224.

[16] H. Aydin, P. Mejía-Alvarez, D. Mossé, and R. Melhem, "Dynamic and Aggressive Scheduling Techniques for Power-Aware Real-Time Systems," in *Proceedings of the 22nd RTSS*, London, UK, 2001, pp. 95–105.

[17] R. Watanabe, M. Kondo, M. Imai, H. Nakamura, and T. Nanya, "Task Scheduling under Performance Constraints for Reducing the Energy Consumption of the GALS Multi-Processor SoC," in *Proceedings of the DATE in Europe*, Nice, France, 2007, pp. 797–802.

[18] R. Ubal, J. Sahuquillo, S. Petit, and P. López, "Multi2Sim: A Simulation Framework to Evaluate Multicore-Multithreaded Processors," in *Proceedings of the 19th SBAC-PAD*, Gramado, Brazil, 2007, pp. 62–68.

[19] J.L. March, J. Sahuquillo, H. Hassan, S. Petit, and J. Duato, "Extending a Multicore Multithread Simulator to Model Power-Aware Hard Real-Time Systems," in *the 10th ICA3PP*, Busan, Korea, 2010, pp. 444–453.

[20] Real-Time Bench, Vasteras, Sweden, *WCET Analysis Project. WCET Benchmark Programs*, 2006, [Online]. Available: http://www.mrtc.mdh.se/projects/wcet/.