https://doi.org/10.17979/spudc.9788497498081.1083

Planificación multinúcleo en sistemas particionados

Vicent Brocal
Fent Innovative Software Solutions (FentISS)

Alfons Crespo, Patricia Balbastre
Universitat Politècnica de València

Resumen

Actualmente, las aplicaciones embebidas se consideran de complejidad creciente y, como consecuencia, de mayor dificultad para su desarrollo y validación. Debido a estas características, existe un gran interés en ejecutar múltiples aplicaciones sobre una única plataforma (mono o multinúcleo). En los sistemas de tiempo real, cada vez más se ejecutan aplicaciones con diferentes niveles de criticidad donde múltiples componentes con diferentes restricciones de tiempo real se integran en una misma plataforma computacional. Al mismo tiempo en el que proliferan los sistemas de criticidad mixta, la plataforma computacional está migrando de mono núcleo a multinúcleo. El presente trabajo describe la planificación de sistemas particionados multinúcleo, desde la asignación de tareas a núcleos hasta la generación de la planificación para cada procesador.

1. Introducción

De entre todos los campos de la informática, los Sistemas de Tiempo Real es donde más beneficios se esperan obtener de la creciente utilización de la tecnología multinúcleo. Pero aprovechar al máximo las capacidades de computación de un sistema multinúcleo en tiempo real es bastante difícil. Esto es principalmente debido al hecho de que los algoritmos de planificación para sistemas multinúcleo son significativamente más complejos que para un sistema mono-núcleo. La teoría que hay sobre este campo, como Leung and Whitehead (1982) o Stankovic et al. (1995), demuestra que la planificación sobre una arquitectura multinúcleo es un problema tipo NP-Completo, pero aparte de aumentar la complejidad, también plantea nuevas posibilidades para las aplicaciones empotradas. Actualmente, las aplicaciones embebidas se consideran de complejidad creciente y, como consecuencia, de mayor dificultad para su desarrollo y validación. Debido a estas características, existe un gran interés en ejecutar múltiples aplicaciones sobre una única plataforma (mono o multinúcleo). Para facilitar este modelo el tiempo de ejecución y el espacio de memoria de cada aplicación deben ser protegidos de otras aplicaciones en el sistema.

En los sistemas de tiempo real, cada vez más se ejecutan aplicaciones con diferentes niveles de criticidad donde múltiples componentes con diferentes restricciones de tiempo real se integran en una misma plataforma computacional [5]. La motivación que hay detrás de hacer convivir aplicaciones de diferente naturaleza son: reducir costes, volumen, peso y consumo eléctrico. Los campos donde se utilizan los sistemas de criticidad mixta son: aviónica [11], espacio[13], automóvil [8], etc.

Al mismo tiempo en el que proliferan los sistemas de criticidad mixta, la plataforma computacional está migrando de mono núcleo a multinúcleo [12], [9]. Se estima que las arquitecturas multinúcleo se utilizarán en el 50 % de las aplicaciones industriales en 2017. Las arquitectura multinúcleo ofrecen una solución muy competitiva en el desarrollo de aplicaciones robustas en el campo de los sistemas de alta criticidad. Sin embargo, suponen una desafío para los desarrolladores de software.

1.1. Sistemas particionados

La industria aeronáutica definió la arquitectura de sistemas particionados con el fin de incrementar la seguridad y predicibilidad de los sistemas de control. Un sistema particionado está configurado por varias aplicaciones independientes llamadas particiones ejecutándose sobre una capa de virtualización. La Figura 1 muestra la arquitectura de un sistema particionado. En la arquitectura de la figura anterior de observa: la capa de virtualización, formada por el hipervisor cuya función es ofrecer una máquina virtual sobre la que se ejecuten las particiones; las particiones que contienen las aplicaciones que van a ejecutarse sobre la máquina virtual. En los sistemas de criticidad mixta, habitualmente las particiones contienen aplicaciones con el mismo nivel de criticidad, de forma que el aislamiento espacial y temporal de la arquitectura particiones hace que si una partición falla, el resto de particiones no se ven afectadas.

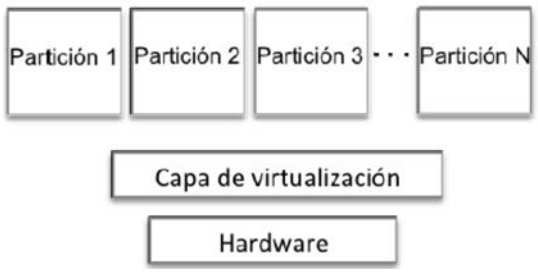

Figura 1: Sistema particionado mononúcleo

2. Multicore en sistemas particionados

De la misma manera que en la versión mononúcleo, un sistema participando multinúcleo está formado por 3 capas (Figura 2). La plataforma hardware, (que en este caso contiene n núcleos), la capa de hipervisor y la capa de particiones.

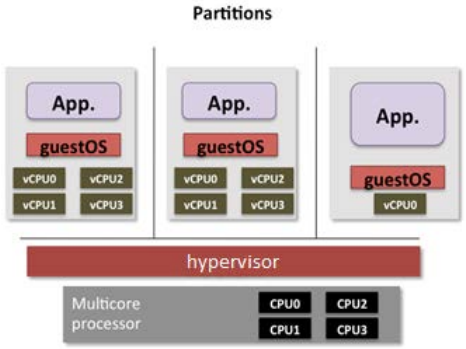

Figura 2: Sistema particionado multinúcleo

3. Generación de la planificación en un sistema particionado multicore

La planificación de sistemas de tiempo real multinúcleo se encarga de dos problemas: asignación de tareas a núcleos y planificación de las tareas.

En [1] se lleva a cabo un estudio sobre los algoritmos de asignación de tareas a núcleos más conocidos llegando a la conclusión de que el WFDU (Worst Fit Decreasing Utilization) consigue un reparto de la carga más homogéneo.

El algoritmo multinúcleo utiliza como base el planificador mononúcleo EEDF ([4]) adaptado para varios procesadores. El modelo de tareas utilizado está compuesto por: particiones, tareas y flujos punto a punto (end-to-end flows, ETEFs).

Antes de pasar a explicar cómo funciona este algoritmo, es necesario introducir la definición de un parámetro que se añadirá al modelo de tareas anterior.

- Interferencia de tarea: Es el porcentaje del WCET de una tarea que supone una interferencia generada por la propia tarea. La interferencia supone un retraso en la ejecución del resto de tareas en otros cores debido al acceso concurrente al bus de memoria y otros periféricos compartidos. Una tarea ejecutándose concurrentemente junto a otras tareas en otros núcleos puede intentar el acceso al bus al mismo tiempo que otras tareas. La tarea a la que se le asigna el bus retrasará la ejecución del resto de tareas en otros núcleos que también quieran acceder al bus en ese momento.

La Figura 3 muestra como se tiene encuentra la interferencia definida en la planificación. En el ejemplo hay 3 núcleos. En los núcleos 0 y 1, hay una tarea que se activa en el instante 0 mientras que en el núcleo 2 hay una tarea que se activa en el instante 5. C_i hace referencia el WCET de cada tarea (coloreado en azul, amarillo o rojo), mientras que I_i es la interferencia de tarea (coloreada en gris) expresada en unidades de tiempo y no en porcentaje para facilitar la comprensión en la representación. Por ejemplo, la tarea en el núcleo 0 sufre 2 unidades de interferencia debido a la ejecución de la tarea en el núcleo 1 (I_1). Más tarde, cuando la tarea en el núcleo 2 se activa, sufre una unidad más de interferencia (I_2). Respecto a la tarea en el núcleo 2, cuando se activa en el instante 5 sufre 2 unidades de interferencia del núcleo 1 (I_1) y una unidad del núcleo 0 (I_0).

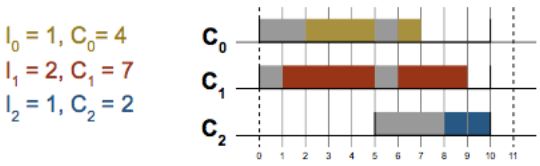

Figura 3: Ejemplo de interferencia entre tareas de diferentes núcleos

Una vez se ha definido el concepto de interferencia, el algoritmo de planificación multinúcleo sigue los siguientes pasos, para cada instante y cada procesador:

1. Seleccionar la siguiente tarea a ejecutar. Para ello se realiza lo siguiente:
   - Se selecciona la tarea con más prioridad. Para ello se siguen las reglas del algoritmo EEDF propuesto en [4].
   - Si la tarea más prioritaria no es la que se está ejecutando actualmente, se calcu-

la la interferencia generada en los otros núcleos debido al cambio de contexto.
- Si este cambio genera más interferencia que mantener la tarea actual ejecutándose, entonces se mantiene la tarea actual en ejecución. Si no, se selecciona la tarea de más prioridad.

2. Se calcula la interferencia generada en todos los núcleos y se añade al tiempo restante de ejecución de cada tarea.

Este algoritmo tiende a retrasar la ejecución de tareas más prioritarias en núcleos donde hay poca carga y tiempo suficiente para ejecutar las tareas cuando no hay tareas ejecutándose en otros núcleos. De esta forma la interferencia es mínima. Esta situación se mostrará en el ejemplo del apartado 4.

4. Ejemplo

Con los requisitos y el diseño especificado en las secciones anteriores, se adaptó la herramienta Xoncrete ([3]) para poder configurar un sistema particionado multinúcleo. Xoncrete es una herramienta que permite especificar completamente la configuración de un sistema participando, en concreto es compatible con el estándar ARINC 653 y es capaz de generar el fichero de configuración de diferentes versiones de XtratuM.

En esta sección se va a presentar un ejemplo para plasmar cómo funciona el algoritmo de planificación multinúcleo propuesto. El usuario define en Xoncrete el modelo temporal mediante su interfaz gráfica. Nos centraremos sólo en el modelo temporal aunque con Xoncrete podemos definir todo el sistema, como recursos hardware, memoria, comunicaciones, etc. El modelo temporal usado como ejemplo consta de 4 particiones y 3 núcleos. Los detalles de los parámetros de las tareas se muestran en la Figura 4.

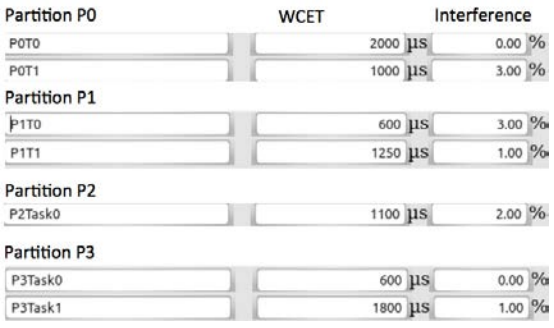

Figura 4: Task parameters of partitions

El modelo consta además de 5 ETEFs y 1 plan que contiene todos los ETEFs definidos. Los parámetros temporales de los ETEFs se muestran en la Tabla 1. Los steps asociados a cada ETEF junto con los graphs de dependencia entre ellos se pueden ver en la Figura 5.

Cuadro 1: Parámetros temporales de los ETEFs (en microsegundos)

ETEF	Periodo min	Periodo max	Plazo
ETEF0Act0	4500	5500	3100
ETEF0Act1	25000	25000	7000
ETEF0Act2	10000	10000	7000
ETEF0Act3	20000	35000	10000
ETEF0Act4	12500	12500	5000

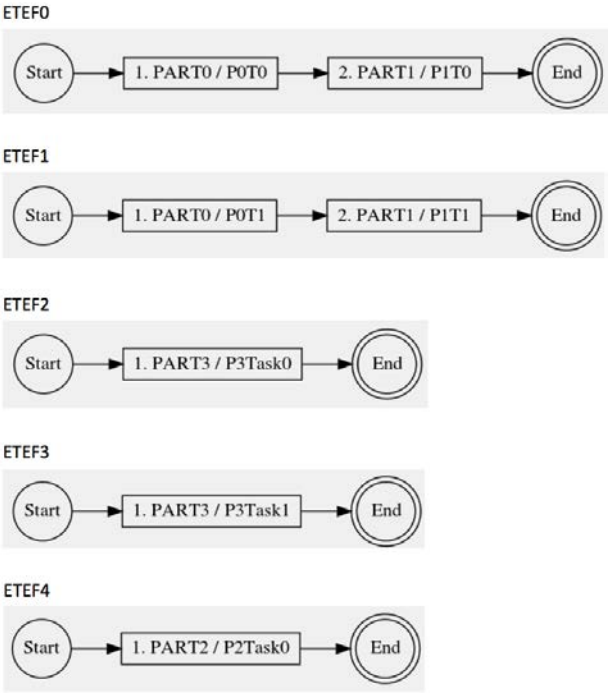

Figura 5: Grafos de dependencia entre steps

Una vez que toda la información sobre el modelo temporal ha sido introducida, se realiza la asirnación de recursos temporales mediante la opción .ªnalysis:Temporal analysis". En este caso, se realizarán dos acciones en orden:

1. Calcular el MAF y los periodos efectivos de los ETEFs. Esto se arealiza en la ventana "Temporal behaviour" mediante los botones "Calculate MAF" y "Calculate periods". Estos cálculos se realizan siguiendo el trabajo publicado en [2]. Los resultados se muestran en la Tabla 2.

2. Generar la planificación: En la ventana "Schedule Generation"se realizan los siguientes pasos de forma transparente al usuario:

Cuadro 2: Periodos de los ETEFs (en microsegundos)

ETEF	Periodo
ETEF0Act0	5000
ETEF0Act1	25000
ETEF0Act2	10000
ETEF0Act3	25000
ETEF0Act4	12500

- Asignación de tareas a núcleos. Se sigue el algoritmo WFDU. El resultado se muestra en la Tabla 3.

Cuadro 3: Asignación de tareas a núcleos

Core name	Tasks
Core 0	P0T0, P0T1
Core 1	P1T0, P1T1
Core 2	P2T0, P3T0, P3T1

- Ejecución de algoritmo de planificación multinúcleo explicado en secciones anteriores.
- Finalmente, se muestra el resultado en el cronograma de ejecución.

Las siguientes figuras (6, 7 y 8) muestran los cronograma de ejecución por núcleo y por ETEF y un resumen de los resultados de utilización. En la Figura 6 se puede ver como en aquellos momentos en los que puede haber una interferencia significativa, el planificador intenta no ejecutar tareas al mismo tiempo en diferentes núcleos. Por tanto, siempre que sea possible y haya suficiente tiempo para ello, los cores no estarán activos al mismo tiempo para evitar las interferencias. Evidentemente esto no será siempre posible ya que de lo contrario no haría falta tener varios núcleos si no van a ejecutarse en paralelo en ningún momento. Pero, el algoritmo de planificación intenta minimizar la interferencia producida.

Respecto a los resultados que aparecen en la Figura 8 el término Useful CPU time se refiere al tiempo utilizado por las tareas sin contar las interferencias entre núcleos y las sobrecargas. Mientras que el término Effective cpu usage se refiere a la cantidad de tiempo total incluyendo interferencias y sobrecargas.

5. Trabajo relacionado

Respecto a la asignación de tareas a núcleos, se puede resolver, entre otras técnicas con el *bin-packing* [7]. El problema es NP-hard ya que conforme crecen el número de tareas y procesadores, el coste computacional crece exponencialmente. Por este motivo no es conveniente utilizar algoritmos óptimos. En [1] se realiza una comparativa de los algoritmos de bin-packing aplicados a asignación multi núcleo.

Una vez se han asignado las tareas, estas deben planificarse. Para ello pueden utilizarse los algoritmo de planificación monoprocesador más habituales. En [6] se realiza un interesante estado del arte en planificación multiprocesador. Respecto a sistemas de criticidad mixta en sistemas multinúcleo, en [10] se proponen dos aproximaciones para la planificación: local y global. En nuestro caso, solo consideramos la local debido al aislamiento que debe haber entre particiones, ya que hay aplicaciones de alta criticidad y el aislamiento es un requisito indispensable.

6. Conclusiones

En este articulo se ha presentado una politica de planificación para sistemas particionados multinúcleo. La metodologia seguida se ha descrito y puesto en práctica mediante un ejemplo. El algoritmo de planificación tiene en cuenta las interferencias generadas entre núcleos por la compartición del bus de memoria. Como trabajo futuro se plantea la consideración de otros recursos compartidos, así como la evaluación de este algoritmo junto con otras propuestas existentes para sistemas particionados de criticidad mixta.

Agradecimientos

Este articulo ha sido subvencionado por el Ministerio de Economía y Competitividad bajo los proyectos TIN2014- 56158-C4-1-P y TIN2014- 56158-C4-4-P.

Referencias

[1] Andrés Martínez, José Enrique Simó, Patricia Balbastre and Alfons Crespo. Análisis y evaluación de políticas de planificación en sistemas particionados multi-núcleo. In *Jornadas de Automatica*, September 2015.

[2] Vicent Brocal and Patricia Balbastre. Selección del periodo para la minimización del hiperperiodo. *Revista Iberoamericana de Automática e Informática Industrial RIAI*, 10(2):186 – 196, 2013.

[3] Vicent Brocal, Miguel Masmano, Ismael Ripoll, Alfons Crespo, and Patricia Balbastre. Xoncrete: a scheduling tool for partitioned real-time systems. In *Embedded Real-Time Software and Systems*, 2010.

[4] Vicent Brocal, Yolanda Valiente, Patricia Balbastre, Alfons Crespo, and Pedro Alber-

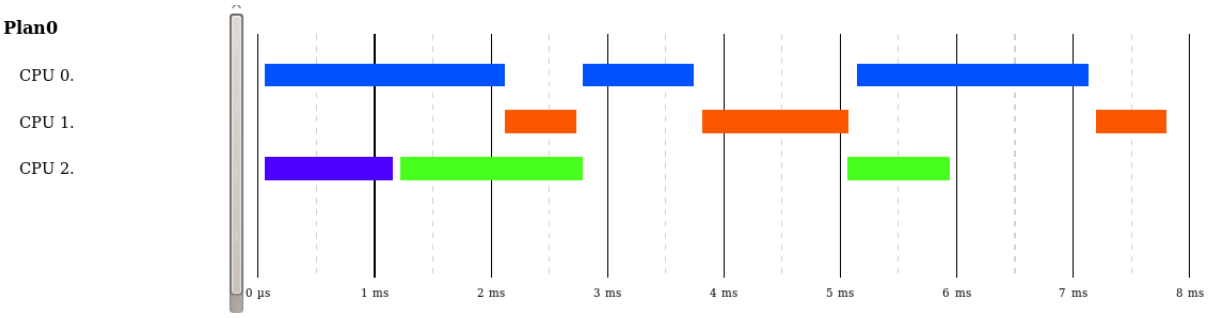

Figura 6: Cronograma de ejecución por núcleo

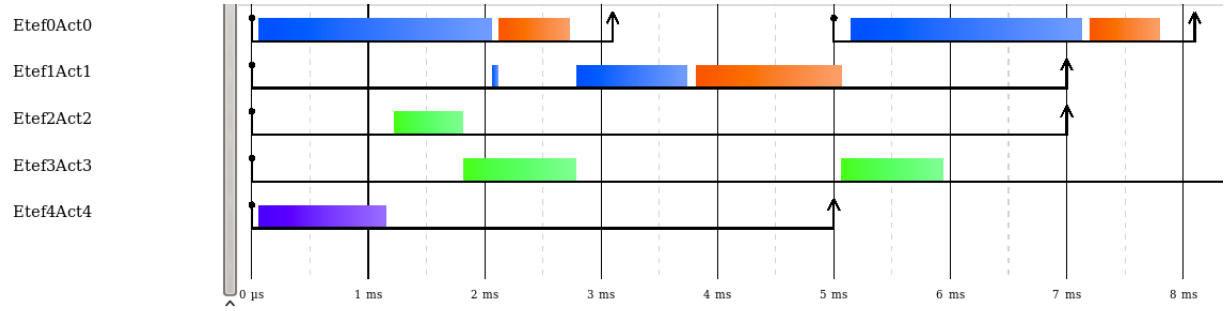

Figura 7: Cronograma de ejecución por ETEF

tos. *Scheduling Algorithms for Cache Effect Optimisationin Partitioned Systems*, pages 491–500. Springer Singapore, Singapore, 2016.

[5] EU Commision. Workshop on Mixed Criticality Systems, 2012. Brussels. cordis.europa.eu/fp7/ict/computing/home en.html.

[6] Robert I. Davis and Alan Burns. A survey of hard real-time scheduling for multiprocessor systems. *ACM Comput. Surv.*, 43(4):35, 2011.

[7] M. R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[8] H. Heinecke, J. Bortolazzi, K.-P. Schnelle, J. l. Mate, H. Fennel, and T. Scharnhorst. AUTOSAR ,Äì An industry-wide initiative to manage the complexity of emerging Automotive E/E-Architectures, 203. http://papers.sae.org/2004-21-0042/.

[9] M. S. Mollison, J. P. Erickson, J. H. Anderson, S. K. Baruah, and J. A. Scoredos. Mixed-criticality real-time scheduling for multicore systems. In *10th International Conference on Computer and Information Technology (CIT)*, pp. 1864,Äì1871, 2010.

[10] Malcolm S. Mollison, Jeremy P. Erickson, James H. Anderson, Sanjoy K. Baruah, and John A. Scoredos. Mixed-criticality real-time scheduling for multicore systems. In *10th IEEE International Conference on Computer and Information Technology, CIT 2010, Bradford, West Yorkshire, UK, June 29-July 1, 2010*, pages 1864–1871, 2010.

[11] John Rushby. Partitioning in avionics architectures: Requirements, mechanisms, and assurance, 1999.

[12] M. Vaidehi and T. R. Gopalakrisgnan. Multicore applications in real time systems. In *Multicore Applications in Real Time Systems*, Vol 1, Issue 1, pp 30-35, 2008.

[13] James Windsor and Kjeld Hjortnaes. Time and space partitioning in spacecraft avionics. *Space Mission Challenges for Information Technology*, 0:13–20, 2009.

Nr etefs	Temporal behaviour		Schedule generation		
	Maf	Utilisation	Deadlines misses	Useful cpu time	Effective cpu usage
5	50000	27 %	0	27.65%	29.36%

Figura 8: Resultados