WCET con cache de instrucciones bloqueable y Lock-MS

Alba Pedro-Zapater[1], Clemente Rodríguez[2], Juan Segarra[1], Rubén Gran[1], Víctor Viñals-Yúfera[1]

[1] Grupo de Arquitectura de Computadores de la Universidad de Zaragoza (gaZ)
Instituto de Investigación en Ingeniería de Aragón (I3A)
Universidad de Zaragoza, Mariano Esquillor s/n, 50018, Zaragoza, Spain.
Tel. +34-976762707, e-mail: albapz@unizar.es
[2]Dpt. Arquitectura y Tecnología de Computadores, U. País Vasco.

Resumen

En sistemas de tiempo real es imprescindible analizar tiempos de ejecución de peor caso (WCET) de tareas, especialmente difícil con memorias cache. Nuestra propuesta automatiza dicho análisis mediante el método Lock-MS para caches de instrucciones bloqueables. Esto permite analizar programas complejos, variando arbitrariamente los parámetros hardware o software relevantes.

Introducción

Los sistemas de tiempo real están incrementando su presencia en la industria y en la vida cotidiana. Podemos encontrar ejemplos en muchos sectores como el aviónico, robótico, automoción, fabricación o control de tráfico aéreo.

Los sistemas de tiempo real estricto (HRTS: *Hard Real Time Systems*) tienen que satisfacer restricciones de tiempo estrictas, que se derivan de la observación, procesamiento y control de su dinámica. Para ello, entre otros aspectos, es necesario determinar cotas superiores ajustadas de los tiempos de ejecución de las tareas que forman la parte software del sistema.

WCET y Caches de Instrucciones

La parte software de un sistema de tiempo real consiste en tareas que desempeñan una determinada funcionalidad. Estas tareas se pueden organizar por prioridades y tienen que ser planificadas de manera que puedan satisfacer sus plazos y períodos. Para poder garantizar una planificación correcta, es necesario analizar el tiempo de ejecución en el peor caso (WCET: *worst-case execution time*) y la forma de secuenciar dichas tareas (planificación). En la Figura 1 vemos una representación de un sistema de tiempo real. En la parte de abajo, en el Análisis, aparece la obtención del WCET para cada tarea, que junto con plazos y períodos permiten el análisis de planificabilidad que determinará si el sistema, con las prioridades asignadas a cada tarea, va a ser capaz de cumplir las restricciones establecidas. Esta información se sustancia en el Software del sistema (Figura 1, arriba, a la derecha), formado por las propias tareas y por el Núcleo del Sistema Operativo de Tiempo Real que se encarga de realizar la planificación de tareas. El Hardware del sistema de tiempo real está formado por el procesador (CPU), las memorias y los periféricos (Figura 1, arriba a la izquierda). Una interrupción de un periférico tiene asociada una tarea que los atiende, y es el planificador quien va a decidir, según la prioridad, si dicha tarea ocupa el procesador o se queda a la espera.

Si los componentes hardware tienen latencia de respuesta fija, el *WCET* de cada tarea se puede calcular a partir de los *WCETs* parciales de cada uno de los bloques básicos de instrucciones que componen dicha tarea. Sin embargo, para mejorar su rendimiento, los procesadores actuales se apoyan en componentes o estructuras como las memorias cache, los predictores de saltos o la organización segmentada del procesador, cuya latencia es variable [1]. Uno de los principales retos para obtener el WCET es el análisis de la jerarquía de memoria [2]. El comportamiento de la memoria cache depende de las referencias pasadas y, en general, es necesario conocer la secuencia de accesos previos para poder predecir la latencia de un determinado acceso. En este trabajo nos centramos en la memoria cache de instrucciones, cuya configuración influye decisivamente en el análisis del WCET (Figura 1, arriba, a la izquierda).

Para evitar la dificultad de predecir de una manera precisa el comportamiento de una cache convencional, se puede usar una cache *bloqueable*. En estas caches el mecanismo de reemplazo está desactivado y los contenidos se almacenan antes de la ejecución de la tarea. Dado que el contenido es conocido y no cambia, el cálculo de fallos y aciertos es trivial y no depende de la secuencia de accesos.

Ahora bien, el desafío en las caches bloqueables es determinar cuál será el mejor contenido a precargar, teniendo en cuenta que el aprovechamiento de la

localidad se puede ver limitado debido a que las líneas seleccionadas de cache van a mantenerse invariables durante toda la ejecución de la tarea.

Se han usado con éxito métodos de análisis *estáticos* para garantizar cotas superiores ajustadas del *WCET* (e.g. [3]). Estos métodos se basan en el análisis de las operaciones del programa y del grafo de flujo de control (secuencias posibles del Contador de Programa, PC), ya sea desde el binario o desde el código fuente. Sin embargo, una de las principales desventajas del análisis estático en la actualidad es la falta de herramientas automáticas.

Módulo Lock-MS

En este trabajo, hemos implementado un módulo de análisis del WCET para la herramienta Otawa que parte del binario del programa. Otawa es una herramienta libre de análisis estático que tiene en cuenta el modelo hardware del procesador y de la memoria para proporcionar una cota ajustada del WCET [4]. Nuestro módulo es capaz de generar las restricciones ILP (*Integer Linear Programming*) del método Lock-MS [3]. Este método ha sido desarrollado por nuestro grupo para determinar el mejor contenido de la cache de instrucciones para minimizar el WCET.

Nuestro módulo, a diferencia de herramientas anteriores, conociendo el número máximo de iteraciones de cada bucle, genera de forma automática las restricciones de Programación Lineal Entera (ILP) que determinan los mejores contenidos de cache y calculan, a la vez, el WCET correspondiente. Esto permite analizar el WCET de un gran número de configuraciones hardware (p.e. asociatividad o tamaño de la cache de instrucciones) o software (p.e. opciones de optimización del compilador), de forma productiva y sencilla. Además nuestro módulo permite analizar programas de gran complejidad.

Conclusiones

Las principales aportaciones de este trabajo son:

- Análisis del *WCET* y optimización con caches de instrucciones bloqueables de manera automática, sin ninguna intervención manual.
- Generación automática del modelo compacto Lock-MS [3], permitiendo que se resuelva en tiempo lineal con respecto al número de condicionales de la tarea (antes exponencial)
- Análisis de *WCET* de benchmarks grandes y complejos.

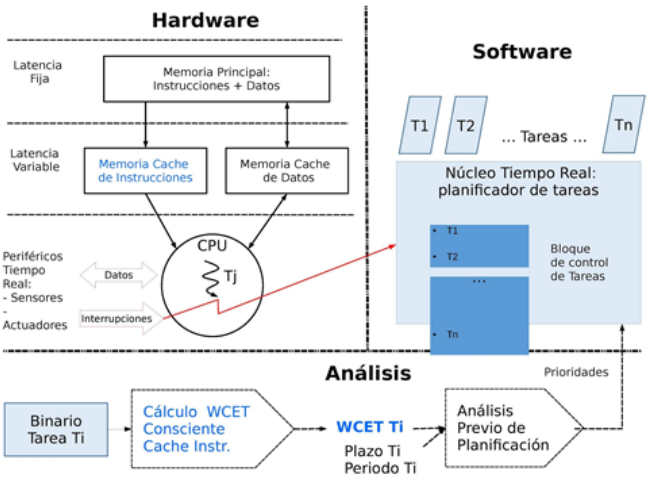

Figura 1. Sistema de Tiempo Real.

Agradecimientos

Este trabajo ha sido parcialmente financiado por los proyectos: TIN2013-46957-C2-1-P, Consolider NoE TIN2014-52608-REDC (Gobierno de España), gaZ: grupo de investigación T48 (Gobierno de Aragón y European ESF), y la beca FPU14/02463.

REFERENCIAS

[1]. WILHELM, R., et al. 2008. The Worst-case Execution-time Problem – Overview of Methods and Survey of Tools. ACM Trans. Embeddded Computing Systems 7, 3. 2008, pp. 36:1–36:53.

[2]. APARICIO, L. C. et al. Avoiding the WCET Overestimation on LRU Instruction Cache. In Proc. *14th IEEE Int. Conf. on Embedded and Real-Time Comp. Syst. and Apps. (RTCSA 08).* Kaohsiung, Taiwan: IEEE Press, 2008, pp.393–398.

[3]. APARICIO, L. C., et al. Improving the WCET computation in the presence of a lockable instruction cache in multitasking real-time systems. *Journal of Systems Architecture 7*. 2011, pp. 695–706.

[4]. BALLABRIGA, C. et al. OTAWA: An Open Toolbox for Adaptive WCET Analysis. In Proc. of the 8th IFIP WG 10.2 Int. Conf. on Software Tech. for Embedded and Ubiquitous. Syst. (SEUS'10). Springer-Verlag, Berlin, Heidelberg: 2010, pp. 35–46. [4].