Caché de último nivel parcialmente compartida basada en distancia

Antonio García-Guirado[1] Ricardo Fernández-Pascual[1] Alberto Ros[1] José M. García[1]

Resumen— **Es común que los multiprocesadores en un chip (CMPs) utilicen una caché de último nivel (LLC) parcialmente compartida ya que esta opción proporciona un equilibrio entre latencia de acceso y aprovechamiento de la capacidad de almacenamiento. En este trabajo proponemos una nueva manera de mapear direcciones de memoria a bancos de LLC que se basa en la distancia entre los bancos y los núcleos que acceden a ellos. Al contrario que los enfoques tradicionales, nuestro mapeo no crea grupos de núcleos que comparten los bancos LLC que forman parte del grupo, sino que cada núcleo accede a un grupo distinto de bancos cercanos, reduciendo la distancia media recorrida en los accesos a la LLC. Los resultados para un CMP de 64 núcleos indican que nuestra propuesta mejora el tiempo de ejecución y el uso de energía de la red un 13 % en comparación con un mapeo tradicional. Además, los costes de implementar esta propuesta en hardware son insignificantes y sus beneficios aumentan a la vez que el número de núcleos.**

I. Introducción

LOS tiled-CMPs (CMPs basados en celdas) son actualmente la mejor manera de mantener viva la ley de Moore gracias a su baja complejidad de diseño comparada con la de otras alternativas y a la facilidad para incrementar el número de núcleos añadiendo más tiles (celdas).

En un tiled-CMP, los bancos del último nivel de caché (LLC) físicamente están distribuidos uniformemente por el chip, aunque lógicamente distintas organizaciones son posibles, desde una LLC totalmente privada hasta una totalmente compartida, para intentar reducir la latencia de la LLC o el número de costosos accesos a la memoria fuera del chip [1], respectivamente. Las organizaciones intermedias [2, 3, 4], en las que se crean clusters de núcleos que comparten sus bancos de LLC (ver Figura 1) proporcionan un buen equilibrio entre estas dos alternativas extremas.

En un CMP con una LLC (parcialmente) compartida, la latencia y la energía necesarias para resolver las peticiones a la LLC representan un porcentaje importante del tiempo de ejecución y la energía consumida por el chip. Tanto la latencia como la energía dependen directamente de la distancia entre el núcleo que accede y el banco de LLC accedido.

La organización tradicional de LLC no tiene en cuenta la distancia entre un núcleo y los bancos que accede. De hecho, existen grandes diferencias entre distintos núcleos del mismo cluster. En promedio, los núcleos en el centro del cluster ven los bancos más cerca que los núcleos en los bordes o esquinas, por lo que éstos últimos salen perjudicados.

[1]Universidad de Murcia, e-mail: {toni, rfernandez, aros, jmgarcia}@ditec.um.es.

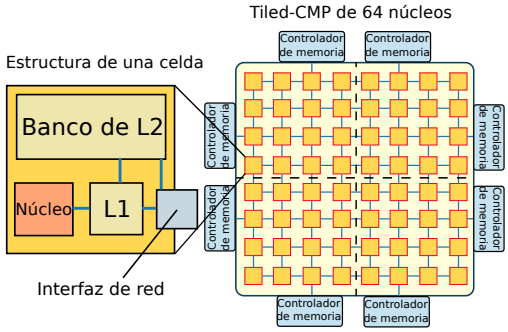

Fig. 1. Diagrama de un tiled-CMP de 64 núcleos con cuatro clusters de 16 núcleos. La líneas de puntos separan los 4 clusters de tiles que comparten sus bancos de LLC.

En este trabajo describimos nuestra propuesta DAPSCO (Distance-Aware Partially Shared Cache Organization) que consiste en usar un mapeo estático distinto para cada núcleo, de manera que cada núcleo acceda a los bancos de LLC que le rodean. El mapeo sigue siendo estático, con lo que los cambios necesarios en el hardware son simples e introducen una sobrecarga insignificante. DAPSCO reduce notablemente la latencia de acceso a la LLC y mejora el tiempo de ejecución (13 %) y el consumo de energía de la red (13 %).

II. Background

Las arquitecturas objeto de nuestra propuesta son los tiled-CMPs como los de la Figura 1. Cada tile contiene un núcleo, cachés L1 privadas de datos e instrucciones, un banco de la LLC (L2) y una interfaz de red para comunicación con otros tiles. Clusters de núcleos comparten sus bancos de LLC. Usaremos el término *grado de compartición* para denotar el número de núcleos que acceden a cada banco de la LLC. Para la coherencia de caché se usa un protocolo MOESI.

El mapeo de bloques de memoria a bancos de LLC se suele realizar usando algunos bits de la dirección ($\log_2$ *gc* bits, donde *gc* es el grado de compartición) como puede verse en la Figura 2. Se utiliza un directorio para mantener la coherencia de caché entre clusters.

A. Coherencia de caché

En una LLC parcialmente compartida son necesarios dos niveles de coherencia: un primer nivel para mantener la coherencia entre las cachés privadas de un cluster, y un segundo nivel para mantener la coherencia entre los bancos de caché de distintos clusters. Asumimos una caché de directorio distribuida por dirección en los bancos de LLC para el primer

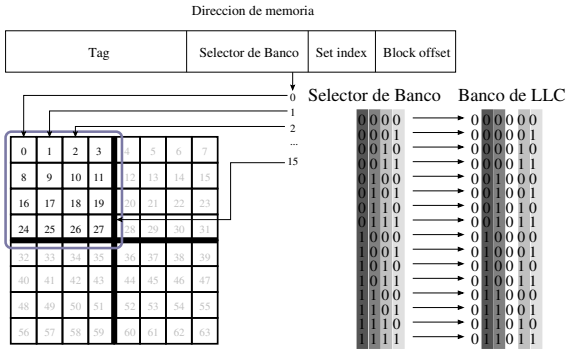

Fig. 2. Seleción de banco en uno de los clusters de una LLC parcialmente compartida con gc de 16. Los 16 núcleos de cada cluster comparten sus bancos de LLC. Los números en los tiles representan sus identificadores.

TABLA I
OVERHEAD DE LA INFORMACIÓN DE COMPARTICIÓN SOBRE LA CAPACIDAD DE CACHÉ DEL CHIP.

Núcleos	Grado de compartición									
	1	2	4	8	16	32	64	128	256	512
64	44 %	22 %	11 %	6.3 %	4.2 %	4.2 %	5.6 %			
128	89 %	45 %	23 %	12 %	7 %	5.6 %	6.9 %	11 %		
256	178 %	89 %	45 %	23 %	13 %	8.3 %	8.3 %	13 %	22 %	
512	356 %	178 %	89 %	45 %	24 %	14 %	11 %	14 %	24 %	44 %

nivel de coherencia y una caché de directorio distribuida entre los controladores de memoria para el segundo nivel. Ocho controladores de memoria se reparten por los bordes del chip y las direcciones de memoria se distribuyen entre ellos.

El directorio necesario para una LLC parcialmente compartida supone una sobrecarga menor que para una LLC privada o totalmente compartida, debido a que las entradas de directorio son más pequeñas. El primer nivel de coherencia necesita entradas con tantos bits como L1s acceden al banco de LLC (es decir, el grado de compartición), y el segundo nivel necesita tantos bits como clusters existen. Sin embargo, una LLC privada o totalmente compartida necesita entradas con tantos bits como núcleos hay en el chip, es decir, entradas de mayor tamaño.

La Tabla I muestra el overhead de los vectores de compartición para distintos números de núcleos y grados de compartición, suponiendo siempre una caché de directorio poco densa con un sobreaprovisionamiento de 4× y bancos de LLC (L2) con un tamaño ocho veces superior al de un banco de L1. El menor overhead suele encontrarse en una organización con grado de compartición intermedio.

B. Mapeo de bloques y distancia a la LLC

La LLC actúa como la última barrera para evitar costosos accesos a memoria principal. Para tener buen rendimiento la tasa de fallos de la LLC debe ser muy baja, por lo que los fallos en L1 deben resolverse obteniendo el dato del banco de LLC deseado. Nuestros experimentos con benchmarks variados muestran que, de media, más del 70 % de los fallos de L1 se resuelven con un acceso a un banco de la LLC dentro del cluster, más de un 10 % con una transferencia caché a caché dentro del cluster, y el resto

3	2.5	2.5	3	3	2.5	2.5	3
2.5	2	2	2.5	2.5	2	2	2.5
2.5	2	2	2.5	2.5	2	2	2.5
3	2.5	2.5	3	3	2.5	2.5	3
3	2.5	2.5	3	3	2.5	2.5	3
2.5	2	2	2.5	2.5	2	2	2.5
2.5	2	2	2.5	2.5	2	2	2.5
3	2.5	2.5	3	3	2.5	2.5	3

Fig. 3. Distancia media en número de enlaces de cada núcleo a los bancos de LLC accedidos en un CMP con 64 núcleos y 4 clusters (grado de compartición de 16).

de los fallos encuentran el dato en otro cluster o en memoria.

Es por ello que la latencia media de los fallos de L1 está determinada principalmente por la distancia entre el núcleo y el banco de LLC del cluster en que se encuentra el dato. Por tanto, el mapeo de bloques a bancos de LLC es clave para mejorar el rendimiento del CMP, ya que determina la distancia de cada núcleo a los bancos de LLC que accede.

Las cachés parcialmente compartidas no hacen nada para reducir esta distancia. Los núcleos se agrupan en clusters dando lugar a accesos a la LLC con latencias largas. Como ejemplo, la Figura 3 muestra la distancia media en número de enlaces de cada núcleo a los bancos de LLC que accede. Conforme nos acercamos a los bordes del cluster la distancia media del núcleo a la LLC aumenta, y esta diferencia se hace mayor cuanto mayor es el grado de compartición.

Este tipo de LLC parcialmente compartida tenía sentido cuando cada cluster de núcleos se encontraba en un chip distinto, ya que acceder a los bancos de LLC dentro del chip era mucho menos costoso. Sin embargo, en un escenario de *many-core* CMPs todos los núcleos se encuentran en el mismo chip, por lo que una organización parcialmente compartida como ésta no optimiza el acceso a los bancos de LLC.

III. DAPSCO: DISTANCE-AWARE PARTIALLY SHARED CACHE ORGANIZATION

Una organización de LLC con grado de compartición g está definida por dos elementos. El primero son las relaciones de acceso entre núcleos y bancos de LLC, accediendo cada núcleo a g bancos de LLC y siendo cada banco accedido a su vez por g núcleos. El segundo elemento es la asignación de una parte igual a $1/g$ del espacio de memoria a cada banco, de forma que varios bancos almacenan bloques de cada una de estas partes. Para que la organización funcione cada núcleo debe ser capaz de acceder a todo el espacio de memoria a través de los bancos de LLC a los que tiene acceso.

La idea principal de DAPSCO consiste en permitir a cada núcleo acceder a un conjunto distinto de los bancos de LLC, de modo que podamos hacer que cada núcleo acceda a bancos cercanos, como puede observarse en la Figura 4. Para conseguir esto, la función de mapeo que determina el banco de LLC que almacena un bloque debe ser distinta en cada

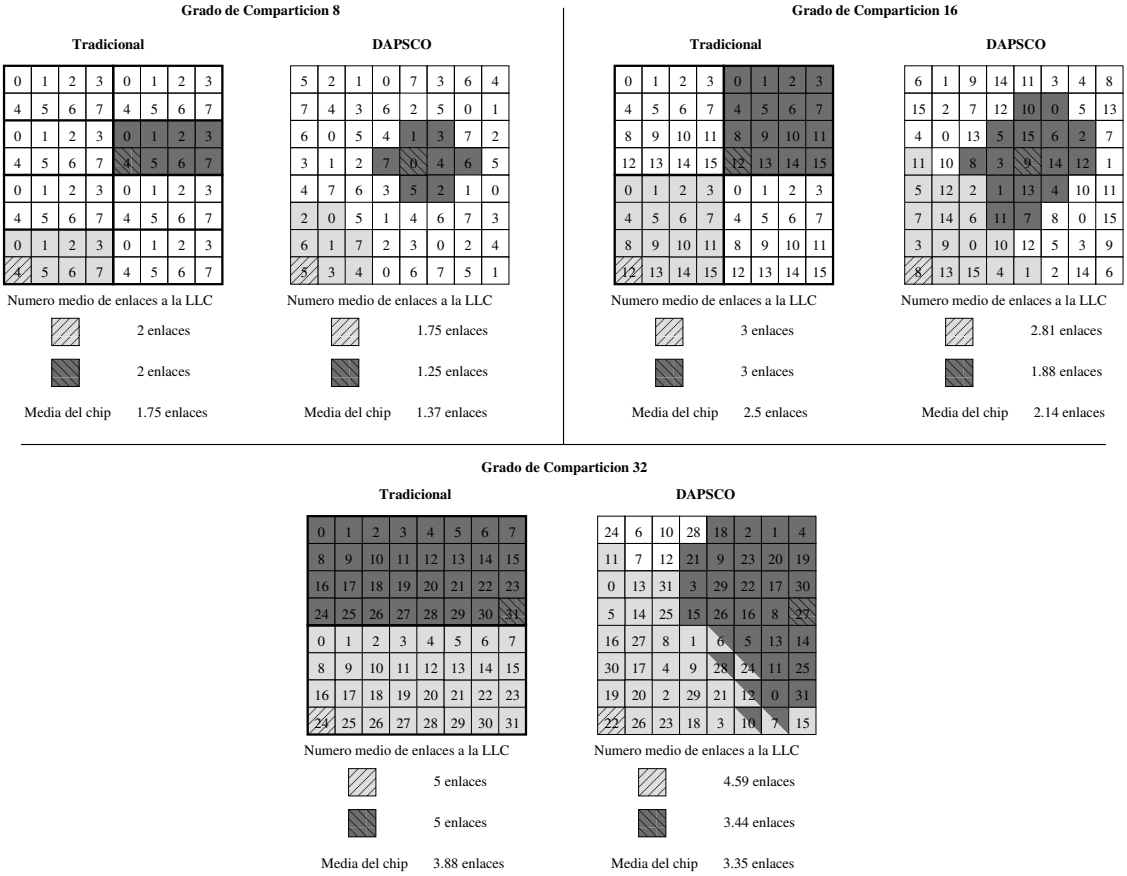

Fig. 4. Bancos de LLC accedidos por los núcleos de una malla. Cada núcleo rayado accede a los bancos sombreados que le rodean. Los números en los tiles representan la porción del espacio de memoria servida por los bancos de LLC. Fíjese en que cada núcleo siempre tiene acceso a todas las porciones del espacio de memoria. DAPSCO reduce notablemente el número de enlaces medios recorridos para acceder a la LLC.

núcleo. Además, todos los núcleos que acceden a un mismo banco deben acceder a la misma porción del espacio de memoria a través de él.

DAPSCO no modifica el protocolo de coherencia ni aumenta la sobrecarga de la información necesaria para mantener la coherencia en comparación con las organizaciones de caché parcialmente compartidas tradicionales, como mostraremos en la Sección III-B.

Encontrar el DAPSCO óptimo para un número de núcleos y un grado de compartición determinados es un problema NP-completo, por lo que en este trabajo usamos algoritmos heurísticos para encontrar configuraciones óptimas o cercanas a la óptima.

A. Explorando el espacio de búsqueda de DAPSCO

Para encontrar el mejor DAPSCO posible tanto para mallas como para toros hemos utilizado dos conocidos algoritmos de optimización global: ascensión de colinas y temple simulado. La metodología explicada en esta sección es aplicable a cualquier topología. Estos algoritmos heurísticos comienzan con la organización de LLC parcialmente compartida tradicional en la que núcleos agrupados en clusters comparten sus bancos de LLC. Los algoritmos generan y evalúan nuevas organizaciones que surgen de la aplicación de los siguientes dos operadores:

- Operador A: dos bancos de LLC que almacenan distintas porciones del espacio de memoria intercambian dichas porciones. Esto también implica que los núcleos que accedían a uno de estos bancos ahora deben acceder al otro para mantener las restricción de que cada núcleo accede a todo el espacio de memoria a través de la LLC.
- Operador B: dos núcleos que acceden a dos bancos distintos de la LLC para la misma porción del espacio de memoria intercambian estos dos bancos.

Es trivial demostrar que cualquier organización de LLC resultante de la aplicación de estos operadores es válida, así como que cualquier organización válida puede obtenerse mediante la aplicación de una secuencia adecuada de estos operadores.

B. Detalles de implementación

Para implementar DAPSCO sólo son necesarios los siguientes cambios en el hardware ya existente:

1. La función que cada núcleo utiliza para mapear direcciones de memoria a bancos de LLC debe representar el mapeo de DAPSCO en lugar del tradicional.
2. La función que mapea los bits del primer nivel de directorio a bancos de L1 debe concordar con las relaciones de acceso de núcleos a bancos de LLC de DAPSCO.

TABLA II
TAMAÑO DE LOS CIRCUITOS DE MAPEO DE DIRECCIÓN A BANCO EN DAPSCO PARA GRADO DE COMPARTICIÓN DE 8.

Núcleos	Número de transistores por tile	Máximo nº de transistores en el camino crítico
64	64	8
128	72	8
256	72	8
512	75	8

3. La función que mapea los bits del segundo nivel de directorio a bancos de LLC debe concordar con las asignaciones de fracciones de espacio de memoria a bancos de LLC de DAPSCO.

Estas funciones se implementan como circuitos combinacionales. Hemos generado el layout de todos estos circuitos tanto para la organización tradicional como para DAPSCO. En los casos (2) y (3) no existe diferencia en el número total de transistores ni en la longitud del camino crítico de estos circuitos entre la organización tradicional y DAPSCO. En el caso (1) la organización tradicional utiliza directamente algunos bits de la dirección de memoria para generar el identificador del banco de LLC a acceder, mientras que en DAPSCO es necesario el uso de un circuito que traduzca estos bits en el identificador. La Tabla II muestra el número de transistores necesarios para realizar esta traducción en DAPSCO en una malla con un grado de compartición igual a 8. El consumo energético y el área ocupada por estos transistores es insignificante. Además, el número de transistores necesarios escala bien con el número de núcleos y la longitud del camino crítico se mantiene constante. La latencia de estos circuitos puede además ocultarse si son accedidos en paralelo a los tags de L1. Como conclusión, las modificaciones en el hardware necesarias para implementar DAPSCO no crean ningún aumento de consumo, área o latencia.

IV. EVALUACIÓN

A. Efectividad de los DAPSCO hallados

Para realizar la búsqueda de DAPSCOs utilizamos un algoritmo de ascensión de colinas y otro de temple simulado, como explicamos en la Sección III-A. Estos algoritmos se ejecutaron repetidamente para varias combinaciones de grado de compartición y número de núcleos, y seleccionamos la mejor solución encontrada para cada una de estas combinaciones. En total, los algoritmos se ejecutaron en un cluster de 268 núcleos (Intel Xeon a 2.33GHz en su mayoría) durante una semana. Los algoritmos se reiniciaban cada vez que la generación de dos millones de nuevas organizaciones consecutivas no mejoraba la mejor solución encontrada por la ejecución actual del algoritmo. Se implementaron algunas optimizaciones que mejoraron la eficiencia de los algoritmos, tales como sólo aplicar operadores que redujeran la distancia media a la LLC de alguno de los núcleos afectados por el operador para dirigir la búsqueda.

La Figura 5 muestra el número medio de enlaces entre los núcleos y la LLC proporcionado por las mejores configuraciones encontradas por los algoritmos, tanto para mallas como para toros. Se muestran tres valores distintos por topología: la organización tradicional, una cota inferior y el mejor DAPSCO encontrado. La cota inferior supone que cada núcleo accede a los bancos de LLC que tiene más cerca, lo cual normalmente no es posible en la práctica, ya que si esto fuera así, los bancos en las esquinas y bordes del chip serían accedidos por menos núcleos que los bancos en el centro, dando lugar a una configuración inválida. Incluimos esta cota para comprobar la efectividad de la búsqueda de DAPSCOs.

Un hecho importante es que, al aumentar el grado de compartición, la distancia con la LLC aumenta y la mejora relativa obtenida con DAPSCO también aumenta. Por tanto, DAPSCO será más beneficioso para el rendimiento y el consumo de energía cuanto mayor sea el grado de compartición ya que las latencias y consumos invariables (como los accesos a los arrays de tags y datos de las cachés) suponen una fracción menor del total y la comunicación con la LLC supone una fracción mayor.

TABLA III
CMP SIMULADO.

Procesadores	64 UltraSPARC-III+ 3 GHz. 2-vías, en orden.
L1 Cache	I&D Separadas. Tamaño: 64KB. Asociatividad: 4-vías. 64 bytes/bloque. Latencia: 1 ciclo.
L2 Cache	Tamaño: 1MB cada banco. 64MB total. Asociatividad: 8-vías. 64 bytes/bloque. Latencia: 2 (tag) + 3 (data) ciclos.
RAM	4 GB DRAM. 8 controladores en los bordes del chip. Latencia: 150 ciclos + retardo on-chip.
Interconexión	Malla bidimensional y toro 8x8. 16 bytes por enlace. Latencia mesh: 2 ciclos/link. Toro: 4 ciclos/link.

B. Metodología de simulación

Para la evaluación hemos usado el simulador GEMS [5] para modelar un tiled-CMP cuyas características pueden verse en la Tabla III, los simuladores Garnet [6] y Orion [7] para modelar la red y su consumo energético, y cargas tanto científicas (lu, lunc, tomcatv, volrend, watersp) como comerciales (apache y jbb).

Hemos evaluado cuatro configuraciones distintas:

- 8Tradicional: organización de caché parcialmente compartida tradicional con grado de compartición de 8.
- 8DAPSCO: DAPSCO con grado de compartición de 8.
- 16Tradicional: organización de caché parcialmente compartida tradicional con grado de compartición de 16.
- 16DAPSCO: DAPSCO con grado de compartición de 16.

Hemos usado dos topologías de red distintas: malla y toro. Añadimos los sufijos malla y toro a los nombres de las cuatro configuraciones anteriores para distinguir la topología utilizada.

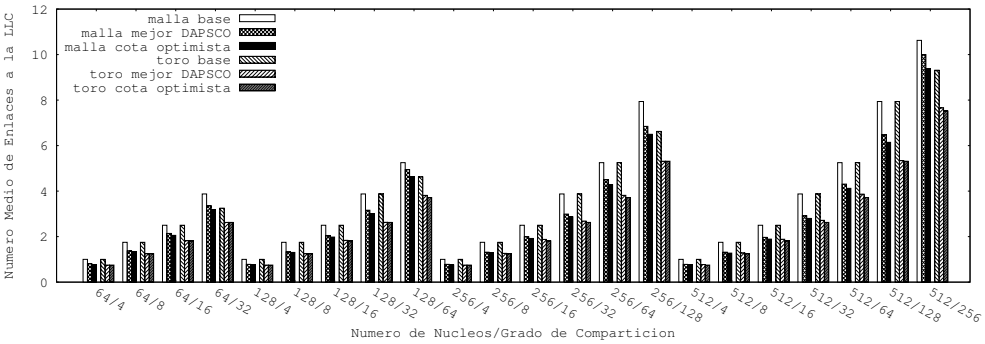

Fig. 5. Número medio de enlaces de los núcleos a los bancos de LLC. El eje x representa el número de núcleos en el chip (de 64 a 512) y el grado de compartición (de 4 a 256).

TABLA IV
DISTANCIA MEDIA DE LOS NÚCLEOS A LOS BANCOS DE LLC.

Configuración	Distancia media a la LLC (enlaces)	Mejora sobre tradicional
8TradicionalMalla	1.75	
8DAPSCOMalla	1.37	24 %
16TradicionalMalla	2.5	
16DAPSCOMalla	2.14	17 %
8TradicionalToro	1.75	
8DAPSCOToro	1.25	40 %
16TradicionalToro	2.5	
16DAPSCOToro	1.88	33 %

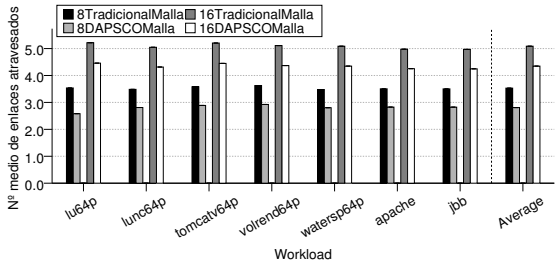

Fig. 6. Número medio de enlaces atravesados para acceder a la LLC. Malla.

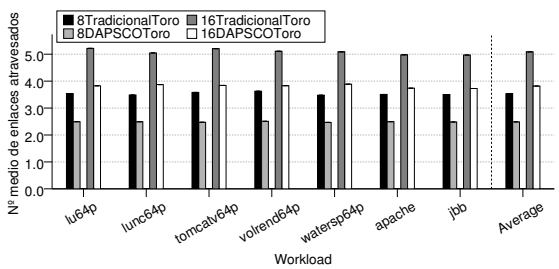

Fig. 7. Número medio de enlaces atravesados para acceder a la LLC. Toro.

El número medio de enlaces para llegar a la LLC en cada configuración se muestra en la Tabla IV.

C. Resultados

La mayor parte de los fallos de L1 se resuelven obteniendo el dato de la LLC. Estos fallos se benefician de la menor distancia a la LLC de DAPSCO. Las Figuras 6 y 7 muestran el número medio de enlaces atravesados para resolver estos fallos usando una malla o un toro. Estos resultados experimentales encajan con los valores medios de DAPSCO mostrados anteriormente en la Tabla IV. La máxima mejora

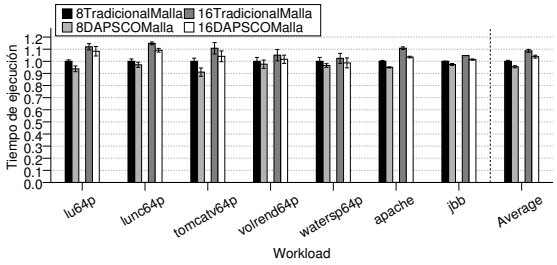

Fig. 8. Tiempo de ejecución normalizado usando una malla.

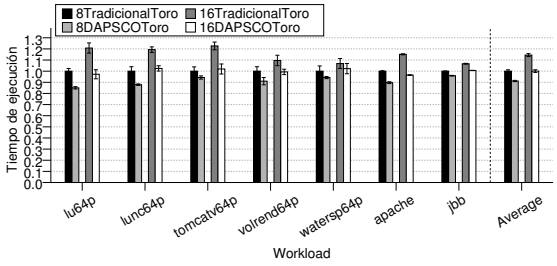

Fig. 9. Tiempo de ejecución normalizado usando un toro.

corresponde al toro y un grado de compartición de 8, con un 40 %.

La reducción en enlaces atravesados se traduce en mejoras en el tiempo de ejecución y en el consumo de energía de la red. Las Figuras 8 y 9 muestran el tiempo de ejecución de las ocho configuraciones evaluadas. Al usar una malla el rendimiento del sistema mejora un 4 % y un 6 % con DAPSCOs para grados de compartición de 8 y 16, respectivamente. Con el toro estas mejoras suben hasta un 10 % y un 13 %, respectivamente. A pesar de que la reducción en número de enlaces es menor con un grado de compartición de 16 que con uno de 8, la ganancia en tiempo de ejecución de DAPSCO es mayor con el de 16 ya que el porcentaje de latencia debido a atravesar la red supone una fracción mayor del tiempo de ejecución cuanto mayor es el grado de compartición.

Las Figuras 10 y 11 muestran el consumo de energía de la malla y del toro. De nuevo, gracias a la reducción en número de enlaces y routers atravesados para acceder a la LLC, la energía se reduce al usar DAPSCO un 4 % y un 6 % con la malla y un 10 % y un 13 % con el toro para grados de compartición de

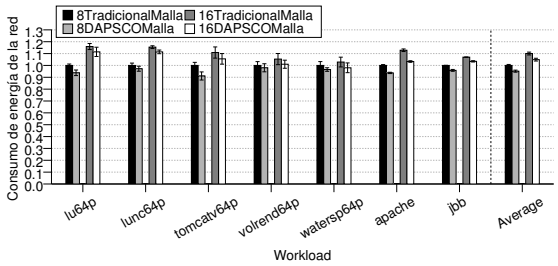

Fig. 10. Consumo de energía normalizado de la malla.

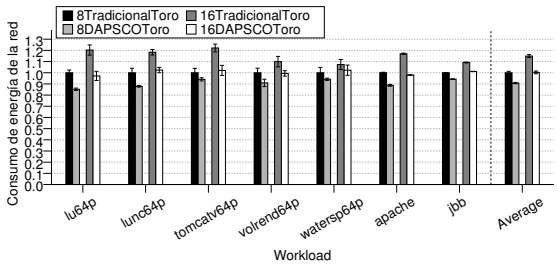

Fig. 11. Consumo de energía normalizado del toro.

8 y 16, respectivamente. Estos resultados muestran la efectividad de DAPSCO.

No obstante, el potencial de DAPSCO aumentará conforme aumente el número de núcleos y el grado de compartición ya que la reducción de enlaces atravesados en el acceso a la LLC tendrá mayor impacto en el tiempo de ejecución y el consumo de la red.

V. Ortogonalidad de DAPSCO

DAPSCO es ortogonal a otras propuestas que utilizan cachés parcialmente compartidas, como Reactive NUCA [8]. Reactive NUCA se basa en clasificar dinámicamente cada bloque en uno de varios tipos de bloque predeterminados (instrucciones, datos privados y datos compartidos), y en aplicar un grado de compartición distinto predeterminado a cada uno de estos tipos de bloque para mejorar el rendimiento.

Hemos aplicado Reactive NUCA a un CMP de 64 núcleos con una malla, y mediante tests exhaustivos hemos determinado que los mejores grados de compartición son: cuatro para instrucciones, uno para datos privados y ocho para datos compartidos. Posteriormente, hemos sustituido la organización de caché parcialmente compartida de Reactive NUCA por DAPSCO. La Figura 12 muestra la comparativa de rendimiento entre Reactive NUCA y Reactive NUCA más DAPSCO. El rendimiento de Reactive NUCA mejora un 3 % de media (hasta un 7 %) al activar DAPSCO.

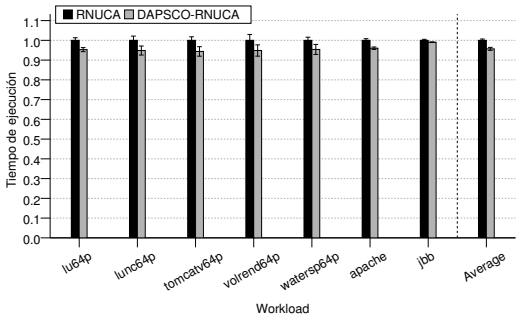

Fig. 12. Rendimiento de Reactive NUCA sin y con DAPSCO.

VI. Conclusiones

Hemos optimizado la organización de caché parcialmente compartida de los CMPs haciendo que cada núcleo acceda a bancos de LLC cercanos. Los costes en hardware de la nueva organización son insignificantes y las mejoras obtenidas en tiempo de ejecución y consumo de energía de la red en comparación con la organización tradicional son notables. Hemos mostrado ejemplos que mejoran el rendimiento de un CMP de 64 núcleos en un 4 % y un 6 % al usar una malla y un 10 % y un 13 % al usar un toro con respecto a la organización tradicional. El consumo de energía de la red también se reduce de manera similar. También hemos mostrado como al aumentar el número de núcleos y el grado de compartición los beneficios son aún más considerables.

Agradecimientos

Este trabajo ha sido financiado por el MEC y la Comisión Europea FEDER mediante los proyectos "Consolider Ingenio-2010 CSD2006-00046" y "TIN2009-14475-C04-02". Antonio García-Guirado también es beneficiario de una beca FPU del MEC (FPU AP2008-04387).

Referencias

[1] Chun Liu, Anand Sivasubramaniam, and Mahmut Kandemir, "Organizing the Last Line of Defense before Hitting the Memory Wall for CMPs," in *Proceedings of the 10th International Symposium on High Performance Computer Architecture (HPCA)*, 2004, pp. 176–185.

[2] B. A. Nayfeh, K. Olukotun, and J. P. Singh, "The Impact of Shared-Cache Clustering in Small-Scale Shared-Memory Multiprocessors," in *Proceedings of the 2nd IEEE Symposium on High-Performance Computer Architecture (HPCA)*, 1996, pp. 74–84.

[3] Jaehyuk Huh, Changkyu Kim, Hazim Shafi, Lixin Zhang, Doug Burger, and Stephen W. Keckler, "A NUCA Substrate for Flexible CMP Cache Sharing," in *Proceedings of the 19th annual international conference on Supercomputing (ICS)*, 2005, pp. 31–40.

[4] Mohammad Hammoud, Sangyeun Cho, and Rami Melhem, "Dynamic Cache Clustering for Chip Multiprocessors," in *Proceedings of the 23rd International Conference on Supercomputing (ICS)*, 2009, pp. 56–67.

[5] Milo M. K. Martin, Daniel J. Sorin, Bradford M. Beckmann, Michael R. Marty, Min Xu, Alaa R. Alameldeen, Kevin E. Moore, Mark D. Hill, and David A. Wood, "Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset," *SIGARCH Comput. Archit. News*, vol. 33, no. 4, pp. 92–99, 2005.

[6] Niket Agarwal, Tushar Krishna, Li-Shiuan Peh, and Niraj K. Jha, "GARNET: A Detailed On-Chip Network Model Inside a Full-System Simulator," in *International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2009, pp. 33–42.

[7] Andrew B. Kahng, Bin Li, Li-Shiuan Peh, and Kambiz Samadi, "ORION 2.0: a Fast and Accurate NoC Power and Area Model for Early-Stage Design Space Exploration," in *Proceedings of the Conference on Design, Automation and Test in Europe (DATE)*, 2009, pp. 423–428.

[8] Nikos Hardavellas, Michael Ferdman, Babak Falsafi, and Anastasia Ailamaki, "Reactive NUCA: Near-Optimal Block Placement and Replication in Distributed Caches," in *Proceedings of the 36th Annual International Symposium on Computer architecture (ISCA)*, 2009, pp. 184–195.