SMPCache: Simulador de Sistemas de Memoria Caché en Multiprocesadores Simétricos

Miguel A. Vega, Raúl Martín, Francisco A. Zarallo, Juan M. Sánchez, Juan A. Gómez

Resumen-- **En este trabajo presentamos un simulador para el análisis y la docencia de sistemas de memoria caché en multiprocesadores simétricos. La simulación se basa en un modelo construido según los principios básicos de la arquitectura de estos sistemas. El simulador, denominado SMPCache, posee una interfaz gráfica completa y funciona en un entorno PC bajo Windows 98 (ó 95). Al principio, el simulador fue concebido como una herramienta para la enseñanza de las memorias cachés en multiprocesadores. Sin embargo, la potencialidad del sistema desarrollado ha probado su utilidad para el análisis de programas y estrategias de diseño de sistemas de memoria en multiprocesadores. Así, el simulador puede ser utilizado para comprender el diseño de organizaciones que ejecuten de forma óptima un determinado tipo de programas paralelos o para mejorar el modo de operar de una arquitectura paralela concreta.**

Palabras clave-- **Sistemas de Memoria Caché, Multiprocesadores Simétricos, Simulador mediante Trazas, Evaluación del Rendimiento, Herramienta de Soporte, Arquitectura Paralela de Computadores**

I. Introducción

LAS cachés son un componente crítico en el rendimiento de cualquier sistema (parte de la bibliografía existente puede encontrarse en [1]).

La forma más frecuente de arquitectura paralela son los multiprocesadores de pequeña a moderada escala que proporcionan un espacio de direcciones físicas global y acceso simétrico a toda la memoria principal desde cualquier procesador, a menudo denominados multiprocesadores simétricos (SMPs) [2]. En la actualidad, entre las categorías de las jerarquías de memoria en multiprocesadores la más extendida es la de memoria compartida por bus. De hecho, muchos de los multiprocesadores existentes ([3]-[8]) se basan en un bus compartido y utilizan un protocolo de espionaje (*snoopy*) para mantener sus cachés coherentes [2],[9].

La simulación mediante trazas suele ser un método efectivo en coste para estimar el rendimiento de los diseños de sistemas informáticos. Especialmente cuando se diseñan cachés, TLBs (*Translation-Lookaside Buffer*), o sistemas de paginación, la simulación por trazas es una forma muy popular de estudiar y evaluar arquitecturas de computadores, obteniendo una estimación aceptable del rendimiento antes de construir el sistema.

En este trabajo presentamos un simulador mediante trazas para sistemas de memoria caché en multiprocesadores simétricos con memoria compartida por bus. Existen simuladores de memoria caché bien conocidos como: TYCHO, DINEROIII ([10], [11]), ACS (*Acme Cache Simulator*), SISMEC [12], bigDIRN,... Partiendo de estos simuladores, hemos desarrollado un nuevo simulador, teniendo en cuenta nuevas consideraciones sobre su capacidad de trabajo en entornos multiprocesador, la potencialidad de análisis del mismo (variabilidad en el proceso de simulación, datos obtenidos y formato, etc.), su interfaz y su portabilidad. Las consideraciones de diseño se han orientado a satisfacer un conjunto de características con fines docentes y de investigación.

En la próxima sección mencionamos las diferentes características que el simulador ofrece al usuario final. En la sección III se explican algunos de los resultados experimentales obtenidos, estudiando su validez. Finalmente, en la última sección se presentan una serie de conclusiones obtenidas del desarrollo y uso del simulador. Las consideraciones teóricas sobre sistemas de memoria caché, y en particular sobre éstos dentro de entornos multiprocesador, aparecen perfectamente desarrolladas en muchos textos de arquitectura de computadores ([2], [9], [13]-[16]), por lo que no las mencionaremos aquí. Todas las operaciones y algoritmos utilizados responden fielmente a lo expuesto en estos textos. Por tanto, los resultados obtenidos con el simulador tienen una traducción casi fidedigna en el mundo real.

II. Características del Simulador

SMPCache 1.0 es una herramienta software para la evaluación de sistemas jerárquicos de memorias cachés en SMPs con memoria compartida por bus, que opera sobre una plataforma PC con sistema operativo Windows 98 (ó 95), y ha sido escrito utilizando un lenguaje visual. SMPCache ofrece una interfaz gráfica típica de Windows, disponiendo además de una ayuda contextual muy completa. La figura 1 da una visión global de su interfaz gráfica.

Departamento de Informática. Universidad de Extremadura
Escuela Politécnica. Campus Universitario s/n. 10071 Cáceres. Spain
E-mail: mavega@unex.es Fax: +34-927-257202

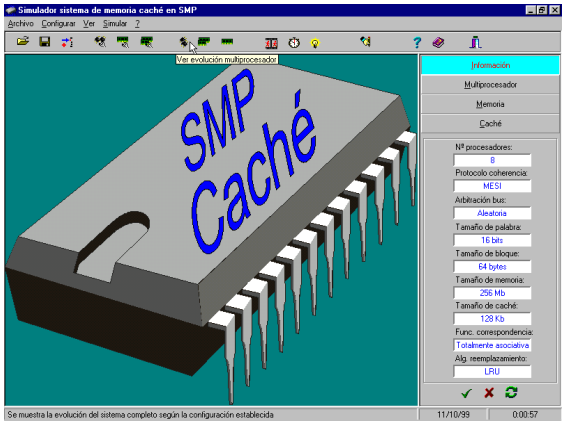

Fig. 1. Interfaz gráfica del simulador.

El simulador permite establecer la configuración de los parámetros que definen la arquitectura del sistema, ofreciéndonos la respuesta del mismo al someterlo a los accesos de memoria generados por los programas (trazas de memoria que se hayan cargado en los distintos procesadores). Por tanto, se trata de una aplicación que podría encargarse de la evaluación de sistemas de memoria en multiprocesadores con fines de investigación.

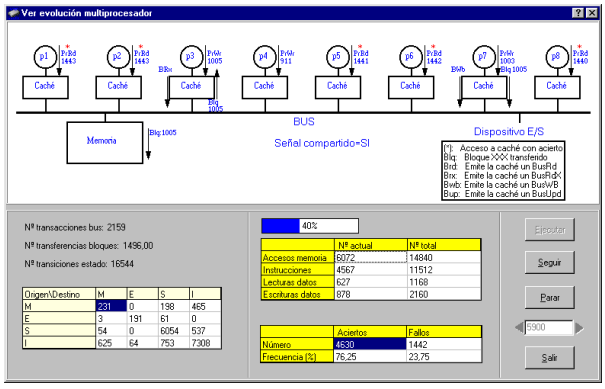

Fig. 2. Visión del multiprocesador en una simulación.

Dada su facilidad de manejo, el simulador también puede ser utilizado igualmente con fines didácticos, ya que permite observar de una forma clara y gráfica cómo va actualizándose el sistema completo a medida que se avanza en la ejecución de los programas (las trazas son leídas). Permitiendo una visión completa de la evolución del multiprocesador (ver figura 2), de una caché en particular o incluso de un determinado bloque de memoria (figura 3). Mostrando, en cada instante, los accesos a memoria realizados por cada procesador, el estado del bus, de cada caché, de cada bloque dentro de cada caché,...

El simulador también permite estudiar el sistema de memoria que mejor se ajusta a nuestras necesidades antes de su implementación física, o simplemente nos ayuda a simular sistemas actuales para ver su eficacia y poder comparar resultados de una forma fácil. Algunos de los factores que se pueden estudiar con el simulador son: Localidad de los distintos programas; influencia del número de procesadores, del protocolo de coherencia caché, del algoritmo de arbitración del bus, del tipo de correspondencia, de la política de reemplazo, del tamaño de la caché (bloques en caché), del número de conjuntos de caché (en correspondencia asociativa por conjuntos), del número de palabras por bloque (tamaño del bloque), del ancho de la palabra,...

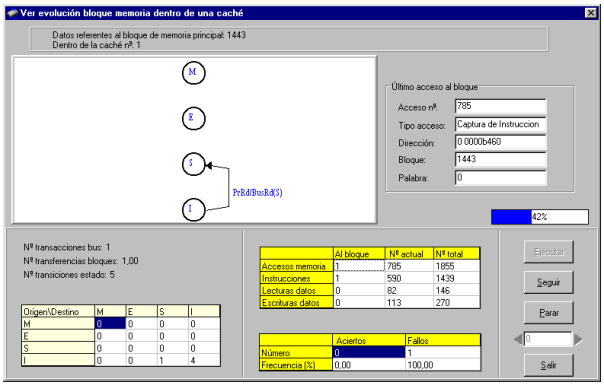

Fig. 3. Diagrama de transición de estados de un bloque de caché concreto durante una simulación.

Además SMPCache presenta mediante datos estadísticos y todo tipo de gráficas (la figura 4 muestra un ejemplo), medidas de interés como:

- Número de transacciones en el bus.
- Nº de transferencias de bloques a través del bus.
- Tráfico en el bus teniendo en cuenta las dos medidas anteriores.
- Número de transiciones de estado (cada bloque en caché tiene un estado asociado).
- Número de transiciones de estado según el estado origen y el destino.
- Accesos a memoria realizados, desglosándolos por tipos: captura de instrucción, lectura de dato o escritura de dato.
- Número de aciertos y fallos de caché, así como frecuencia de aciertos y fallos.

Todos estos datos se presentan a distintos niveles, aunque siempre teniendo en cuenta la interrelación de todos los elementos del sistema. Podemos, por tanto, realizar una simulación observando el multiprocesador completo, y todos los bloques de memoria o un único bloque. También podemos observar una caché específica, y todos los bloques de memoria o un bloque concreto.

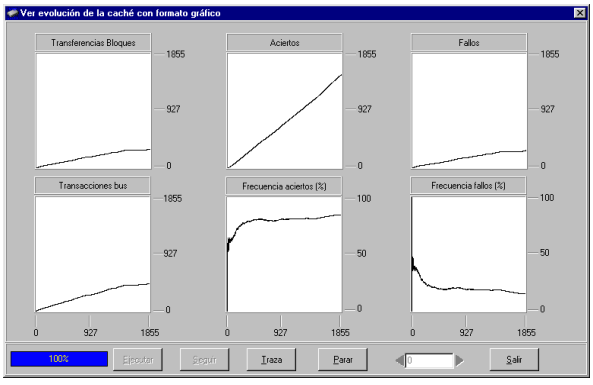

Fig. 4. Datos en formato gráfico de una caché.

La simulación realiza una aproximación programada de las operaciones que realizarían los componentes del sistema de memoria caché en un multiprocesador real. Para ello, se realizan los cálculos adecuados y, al mismo tiempo, se van ofreciendo los resultados, tanto actuales como acumulados. Exiten tres tipos de simulación (paso a paso, con punto de interrupción y ejecución completa), admitiéndose el cambio de un tipo de simulación a otro en mitad de la misma. También existe la posibilidad de abortar la simulación en cualquier instante, con el objeto de corregir algún detalle de la organización.

A. *Organización Hardware*

En cuanto a la organización de la jerarquía de memoria y del multiprocesador, el simulador ofrece las posibilidades resumidas en la tabla I. Además, se ha diseñado para que se pueda ampliar con facilidad (p.e. añadir un nuevo protocolo de coherencia).

TABLA I

CARACTERÍSTICAS SOPORTADAS POR EL SIMULADOR.

Procesadores en el SMP	1, 2, 3, 4, 5, 6, 7 u 8
Protocolos de coherencia caché	MSI, MESI o DRAGON
Algoritmos arbitración del bus	Aleatorio, LRU o LFU
Anchos de palabra (en bits)	8, 16, 32 ó 64
Palabras en un bloque	1, 2, 4, 8, 16, 32, 64, 128, 256, 512 ó 1024
Bloques en memoria principal	1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, 8192, 16384, 32768, 65536, 131072, 262144, 524288, 1048576, 2097152 ó 4194304
Bloques en caché	1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024 ó 2048
Funciones de correspondencia	Directa, asociativa por conjuntos o totalmente asociativa
Conjuntos de caché (para asociativa por conjuntos)	1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024 ó 2048
Algoritmos de reemplazo	Aleatorio, LRU, FIFO o LFU
Políticas de escritura	Post-escritura (por protocolos de coherencia utilizados)
Niveles de caché	1
Referencias	A palabras de memoria
Tamaño de bloque máximo	8 Kb
Tamaño de memoria principal máximo	32 Gb
Tamaño de caché máximo (sin etiquetas, bits de estado, campos de cómputo, ...)	16 Mb

Todos estos parámetros de configuración guardan entre sí las relaciones adecuadas según los modelos teóricos. El simulador avisa cuando se selecciona un valor que entra en conflicto con otros valores previamente seleccionados para otros parámetros.

Las distintas selecciones que se hacen en el simulador para configurar una organización determinada pueden guardarse en ficheros ASCII de datos para una futura carga, evitando así la realización de numerosas selecciones para configurar la misma arquitectura. Más aún, es posible establecer una configuración inicial por defecto para el simulador. Estas características permiten construir una base de datos con las diferentes arquitecturas de memoria, pudiendo emular arquitecturas como Silicon Graphics, Sequent Symmetry, PowerPC, Sun, y otras.

B. *Trazas de Memoria*

Disponemos de un amplio repertorio de trazas de memoria. Muchas de estas trazas provienen de tests realizados con benchmarks y programas de aplicación sobre distintas arquitecturas reales (uniprocesador y multiprocesador) en el *Parallel Architecture Research Laboratory* (PARL), *New Mexico State University* (NMSU), y disponibles mediante ftp anónimo en *tracebase.nmsu.edu*. Las trazas se encuentran en distintos formatos: Dinero (desarrollado por Mark Hill en U.C. Berkeley), PDATS [17], o el formato canónico para trazas multiprocesador desarrollado por Anant Agarwal.

III. RESULTADOS EXPERIMENTALES

Para validar el simulador hemos comparado los resultados obtenidos con los que se obtienen en tests reales, y mostrados en las gráficas de algunos textos [2], [9], [10], [13], [18]. Los resultados que ofrecemos son sólo parte de los experimentos realizados, y corresponden a arquitecturas determinadas con unas trazas concretas.

A. *Trazas Uniprocesador*

En primer lugar, validamos los algoritmos básicos que aparecen en todo sistema de memoria caché, uniprocesador o multiprocesador. Para ello, configuramos el simulador con un único procesador y utilizamos trazas uniprocesador basadas en los primeros millares de accesos a memoria de algunos benchmarks del *SPEC'92 (hydro, nasa7, cexp, mdljd, ear, comp, wave, swm y ucomp)*, según tests reales efectuados sobre un sistema MIPS R2000. Las trazas utilizadas representan una amplia variedad de programas de aplicación "reales".

Se ha realizado un amplio conjunto de experimentos, estudiando parámetros de interés como la localidad de distintos programas, la influencia en la tasa de fallos del tamaño de caché, de la correspondencia, del algoritmo de reemplazo, del tamaño del bloque, etc. Por motivos de espacio sólo mostramos los datos de dos de ellas.

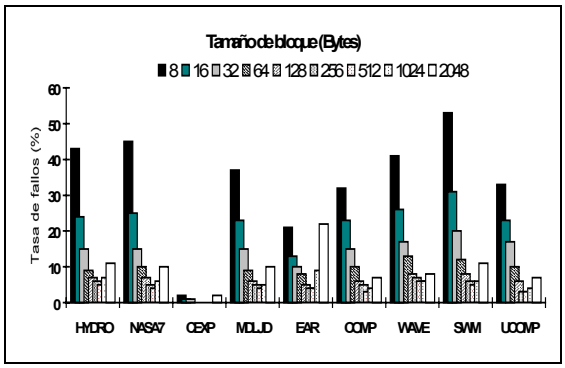

Fig. 5. Tasa de fallos frente a tamaño de bloque.

En la figura 5 estudiamos la influencia del tamaño de bloque en la tasa de fallos. Utilizamos para ello una arquitectura con palabras de 16 bits, caché totalmente asociativa y reemplazo LRU. Los resultados mostrados son consistentes con la teoría. Aumentando el tamaño de bloque disminuye la tasa de fallos, hasta llegar al punto de polución.

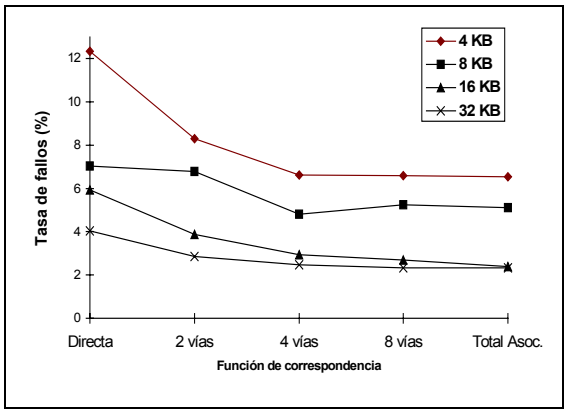

Fig. 6. Tasa de fallos frente a función de correspondencia para distintos tamaños de caché.

Hemos recogido en la figura 6 los resultados obtenidos del benchmark *ear*, en los que comparamos la tasa de fallos para correspondencias diferentes (con reemplazo LRU para el caso de correspondencia asociativa), utilizando distintos tamaños de caché. En este caso la arquitectura posee bloques de 64 palabras de 32 bits. La gráfica indica que en general la tasa de fallos mejora cuando aumenta la asociatividad, aunque al aumentar la caché los beneficios son cada vez menos significativos, lo cual coincide con la teoría.

B. Trazas Multiprocesador

Para este segundo conjunto de experimentos consideramos trazas multiprocesador que poseen decenas de millones de accesos a memoria (referencias) para cuatro benchmarks (*FFT*, *Simple*, *Speech* y *Weather*). Se trata de trazas que representan varias aplicaciones paralelas reales (en [19] se explican las aplicaciones paralelas que dieron lugar a las trazas *FFT*, *Simple* y *Weather*). La tabla II resume los datos principales de cada traza. *FFT*, *Simple* y *Weather* se generaron utilizando el esquema "post-mortem" implementado por Mathews Cherian y Kimming So en IBM. Kirk Johnson y David Kranz (ambos del MIT) son los responsables de *Speech*.

TABLA II

TRAZAS MULTIPROCESADOR UTILIZADAS.

Nombre	Referencias	Lenguaje	Comentarios
FFT	7.451.717	Fortran	Aplicación paralela que simula la dinámica de fluidos utilizando la FFT
Simple	27.030.092	Fortran	Versión paralela de la aplicación SIMPLE
Speech	11.771.664	---	---
Weather	31.764.036	Fortran	Aplicación paralela utilizada para el pronóstico del tiempo, y procedente del *NASA Space Flight Center, Greenbelt Md.*

Utilizaremos una arquitectura multiprocesador compuesta por 8 procesadores con protocolo de coherencia caché MESI (o Illinois [2], [15], [16], [20]), palabras de 16 bits y bloques de 64 bytes. Todas las cachés han sido configuradas con asociatividad de 4-vías y reemplazo LRU.

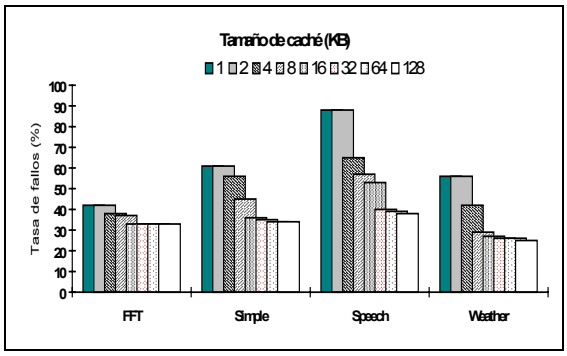

Fig. 7. Tasa de fallos frente a tamaño de caché.

La figura 7 presenta la tasa de fallos cuando se va modificando el tamaño de la caché. Como vemos, la tasa de fallos global del sistema disminuye al aumentar el tamaño de la cachés, lo cual es correcto ya que disminuyen los fallos de capacidad y de conflicto. Para tamaños grandes de caché la tasa de fallos se estabiliza, lo que indica que esos fallos son los llamados forzosos y de coherencia[1], que son independientes del tamaño de la caché. Destacar que generalmente los programas paralelos manifiestan una localidad espacial y temporal menor que la de los programas secuenciales. Así, es normal que las tasas de fallos para trazas multiprocesador sean de mayor magnitud que las de trazas uniprocesador.

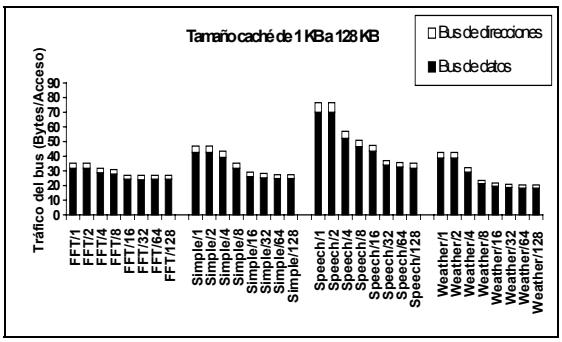

Fig. 8. Tráfico del bus frente a tamaño de caché. El tráfico se divide en tráfico de datos y de direcciones (incluyendo comandos).

La figura 8 muestra el tráfico en el bus del multiprocesador por acceso a memoria para esta misma experimentación. Las conclusiones obtenidas son similares a las obtenidas para la tasa de fallos. El tráfico del bus disminuye al disminuir la tasa de fallos, debido a dos causas principalmente. Por un lado, se producen menos transferencias de datos desde la memoria

[1] Hill [21] propuso las "tres C" (*Compulsory*, *Capacity*, *Conflict*) para los fallos de caché. En multiprocesadores aparecen los fallos de coherencia (*Coherence*), dando lugar a las "cuatro C" [2].

principal compartida a las cachés. Por otro, al existir menos fallos son necesarias menos transacciones en el bus para gestionar el protocolo de coherencia caché.

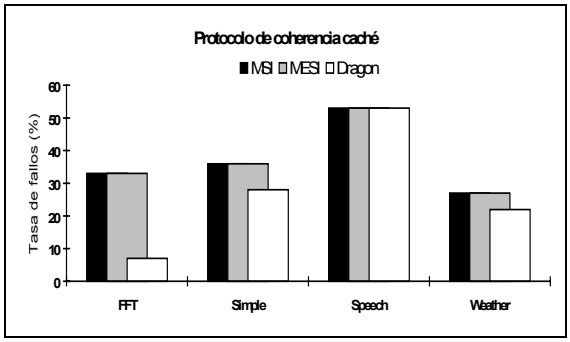

Fig. 9. Tasa de fallos frente a protocolo de coherencia.

Las figuras 9 y 10 estudian la influencia del protocolo de coherencia caché en la tasa de fallos y el tráfico del bus. Seguimos manteniendo la arquitectura anterior. Como vemos, la tasa de fallos para los protocolos MSI ([2], [15], [16]) y MESI (o Illinois) es la misma. Este dato es correcto ya que el protocolo MESI simplemente es una mejora del protocolo MSI (añade el estado E, "Exclusivo") para reducir el número de transacciones en el bus [2]. Observando la figura 10 llegamos a la conclusión de que aunque la tasa de fallos es idéntica para ambos protocolos no ocurre lo mismo con el tráfico en el bus. El protocolo MESI es algo más eficiente, lo cual era de esperar. También se observa que el protocolo Dragon ([2], [6], [15], [16]) genera una menor tasa de fallos y tráfico en el bus que los otros dos. El motivo es que se trata de un protocolo de actualización y no de un protocolo basado en invalidación como los protocolos MSI y MESI. Por tanto, al no existir bloques invalidados (que podrían volver a referenciarse posteriormente) por operaciones de memoria en otros procesadores, disminuyen los fallos de coherencia, y en general la tasa de fallos. Por otro lado, la transacción en el bus para actualización de datos en caché en el protocolo Dragon se ha optimizado para que únicamente se transfiera la palabra modificada, mientras en los protocolos MSI y MESI siempre se transfiere el bloque de memoria completo. De todas formas, existen otras situaciones en las que un protocolo de invalidación es mejor que uno de actualización.

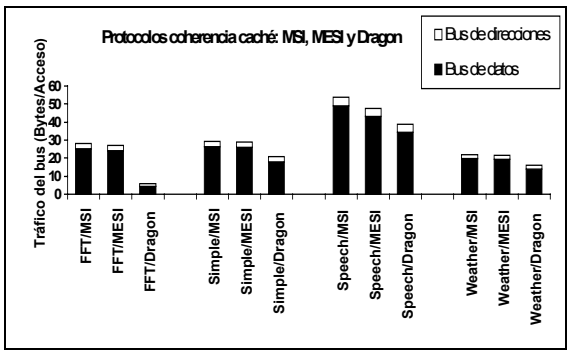

Fig. 10. Tráfico del bus para los distintos protocolos de coherencia.

Finalmente, vamos a considerar las tres trazas de memoria obtenidas utilizando el esquema "post-mortem" (*FFT*, *Simple* y Weather). Las figuras 11 y 12 muestran la influencia del número de procesadores en la tasa de fallos y el tráfico en el bus. Se ha mantenido la arquitectura anterior.

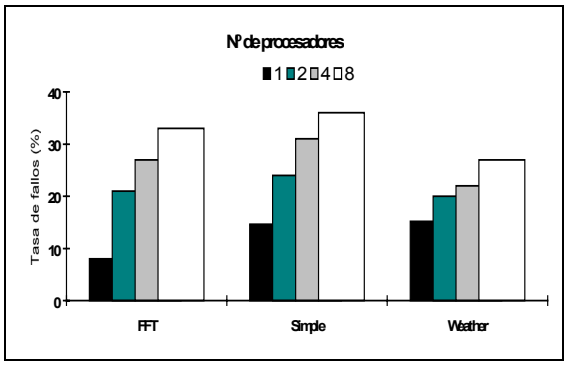

Fig. 11. Tasa de fallos frente a número de procesadores.

Observando ambas figuras se llega a la conclusión de que cuanto mayor es el número de procesadores que intervienen en el procesamiento de una aplicación paralela mayor es la tasa de fallos y el tráfico en el bus. Esto tiene sentido ya que para un protocolo de invalidación como MESI cuanto más cachés puedan poseer el mismo dato más opciones habrá de que alguna de ellas invalide el dato en las demás, generando por tanto, nuevos fallos no esperados y debidos a la coherencia (fallos de coherencia). Este aumento en la tasa de fallos hará aumentar el número de transferencias de bloques. Por otra parte, a mayor número de procesadores será necesario un mayor número de transacciones en el bus para mantener la coherencia de caché. En resumen, cuando se incrementa el número de procesadores para un problema dado, el "conjunto de trabajo" (*working set* [22]) comienza a caber en la caché, y el dominio de los fallos locales (sobre todo de capacidad) en la tasa de fallos se reemplaza por el dominio de los fallos de coherencia.

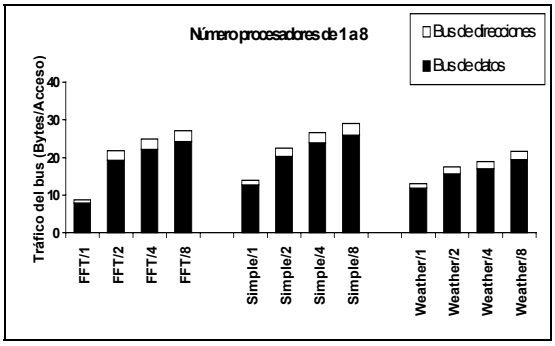

Fig. 12. Tráfico del bus frente a número de procesadores.

IV. Conclusiones y Líneas Futuras

En este trabajo hemos mostrado las principales características de SMPCache, un simulador de sistemas de memoria caché en multiprocesadores simétricos. La experiencia más inmediata nos ha dado muestra de los beneficios del simulador con fines docentes. Los alumnos lo han utilizado para comprobar, experimentalmente y de forma gráfica, los distintos

aspectos teóricos sobre memorias caché y sistemas multiprocesador tratados en los cursos de arquitectura de computadores, con lo cual los conceptos se afianzan de una forma más segura.

Con fines de investigación, los profesores de nuestro entorno académico lo están utilizando por ser una herramienta atractiva y sencilla con la que estudiar de forma satisfactoria configuraciones de memoria en multiprocesadores simétricos que tengan un mejor rendimiento según los programas evaluados. Esperamos que en breve el simulador nos sirva para obtener nuevas e interesantes conclusiones sobre arquitecturas servidor multiprocesador. También estamos trabajando en la incorporación de nuevas características al simulador, tales como tiempo medio de accesos, penalización de fallos,...

Finalmente, para contribuir a un mejor conocimiento de las cachés en sistemas multiprocesador, el simulador está disponible de forma gratuita, con fines docentes y de investigación. Para ello, sólo es necesario contactar con los autores.

V. REFERENCIAS

[1] A. J. Smith, "Bibliography and Readings on CPU Cache Memories and Related Topics", Computer Architecture News, pg. 22-42, Enero 1986.

[2] D. E. Culler, J. P. Singh y A. Gupta, "Parallel Computer Architecture. A Hardware/Software Approach", Morgan Kauffmann, 1999.

[3] Encore Computer Corp., "Multimax Technical Summary", 1986.

[4] T. Lovett y S. Thakkar, "The Symmetry Multiprocessor System", Proc. Int. Conf. Parallel Processing, vol. I, pg. 303-310, Agosto 1988.

[5] L. Monier y P. Sindhu, "The Architecture of the Dragon", Proc. 30th IEEE Int. Conf., IEEE, pg. 118-121, Febrero 1985.

[6] E. M. McCreight, "The Dragon Computer System: An Early Overview", Technical Report, Xerox Corporation, Septiembre 1984.

[7] C. P. Thacker, L. C. Stewart y E. H. Satterthwaite Jr., "Firefly: A Multiprocessor Workstation", IEEE Transactions on Computers, vol. 37, nº 8, pg. 909-920, Agosto 1988.

[8] F. Baskett, T. Jermoluk y D. Solomon, "The 4D-MP Graphics Superworkstation: Computing + Graphics = 40 MIPS + 40 MFLOPS and 100.000 Lighted Polygons per Second", Proc. 33rd IEEE Computer Society Int. Conf. - COMPCOM 88, pg. 468-471, Febrero 1988.

[9] D. Sima, T. Fountain y P. Kacsuk, "Advanced Computer Architectures. A Design Space Approach", Addison-Wesley, 1998.

[10] M. D. Hill y A. J. Smith, "Evaluating Associativity in CPU Caches", IEEE Transactions on Computers, vol. 38, nº 12, pg. 1612-1630, Diciembre 1989.

[11] M. D. Hill, "DineroIII Documentation", página de manual (man) en estilo Unix no publicada, Univ. de California, Berkeley, Octubre 1985.

[12] J. A. Gómez, J. M. Sánchez y M. A. Vega, "Simulación de Configuraciones de Memorias Caché Multinivel con Fines Didácticos", IV Jornadas de Informática, Las Palmas de Gran Canaria, pg. 117-126, Julio 1998.

[13] J. L. Hennesy y D. A. Patterson, "Computer Architecture. A Quantitative Approach", 2ª edición, Morgan Kauffmann, 1996.

[14] K. Hwang y F. A. Briggs, "Computer Architecture and Parallel Processing", McGraw-Hill, 1984.

[15] J. Archibald y J.-L. Baer, "Cache Coherence Protocols: Evaluation Using a Multiprocessor Simulation Model", ACM Transactions on Computer Systems, vol. 4, nº 4, pg. 273-298, Noviembre 1986.

[16] S. Eggers, "Simulation Analysis of Data Sharing in Shared Memory Multiprocessors", Ph. D. Thesis, Univ. de California, Berkeley, Computer Science Division, Technical Report UCB/CSD 89/501, Abril 1989.

[17] E. E. Johnson y J. Ha, "PDATS: Lossless Address Trace Compression for Reducing File Size and Access Time", Proc. IEEE International Phoenix Conference on Computers and Communications, pg. 213-219, Abril 1994.

[18] A. J. Smith, "Line (Block) Size Choice for CPU Cache Memories", IEEE Transactions on Computers, vol. 36, nº 9, pg. 1063-1075, Septiembre 1987.

[19] F. Darema, A. Karp y P. Teller, "Applications Survey Reports-I", IBM RC 12743, Mayo 1987.

[20] M. Papamarcos y J. Patel, "A Low Overhead Coherence Solution for Multiprocessors with Private Cache Memories", Proc. 11th Annual Symposium on Computer Architecture, pg. 348-354, Junio 1984.

[21] M. D. Hill, "Aspects of Cache Memory and Instruction Buffer Performance", Ph. D. Thesis, Univ. de California, Berkeley, Computer Science Division, Technical Report UCB/CSD 87/381, Noviembre 1987.

[22] P. J. Denning, "The Working Set Model for Program Behavior", Communications of the ACM, vol. 11, nº 5, pg. 323-333, 1968.