Reducción del Coste del Reemplazo en los Protocolos COMA

Diego R. Llanos Ferraris, Benjamín Sahelices Fernández y Agustín de Dios Hernández

Resumen— **Se han propuesto hasta la fecha diferentes soluciones al problema del reemplazo en los protocolos COMA. En el presente trabajo se realiza una comparativa de dichas propuestas, utilizando para ello un simulador basado en ejecución que utiliza el protocolo VSR-COMA para el mantenimiento de la coherencia entre las diferentes memorias de atracción. Se describe tanto el mecanismo de simulación empleado como el modelo analítico que permite obtener a partir de éste los índices de rendimiento. Los resultados se presentan a través de curvas de aceleración para diferentes aplicaciones del repertorio Splash-2 utilizando VSR-COMA y las estrategias de reemplazo descritas. Nuestros resultados muestran que el mecanismo de reemplazo propuesto para VSR-COMA permite obtener una mejora en el rendimiento en comparación con los mecanismos propuestos hasta la fecha.**

Palabras clave— **Arquitecturas DVSM, arquitecturas COMA, mecanismos de reemplazo.**

I. Introducción

EL protocolo VSR-COMA [1], [2] es un protocolo de coherencia diseñado para su utilización en un sistema débilmente acoplado formado por estaciones de trabajo conectadas a través de un bus común. VSR-COMA utiliza un mecanismo de gestión de memoria de tipo COMA [3], que supone la existencia en cada nodo de proceso de una memoria que contiene una parte del espacio compartido de direcciones, y que recibe el nombre de "memoria de atracción" (AM) [4]. Las diferentes AMs no contienen una parte concreta del espacio de direcciones, sino que los datos van migrando de la AM de un nodo a la AM de otro a medida que progresa la ejecución y los datos son requeridos en los diferentes nodos. En este sentido, las memorias de atracción actúan como grandes caches que almacenan la parte del espacio de direcciones que cada nodo utiliza en cada momento. Al igual que sucede en las caches convencionales, los datos se transfieren entre las diferentes AM utilizando bloques de tamaño fijo como unidad de coherencia.

VSR-COMA incorpora todas las funcionalidades necesarias para su utilización en condiciones reales de ejecución: asociatividad por conjuntos de las AM, operaciones atómicas que facilitan la sincronización de procesos y un elaborado mecanismo de reemplazo de bloques. Dicho mecanismo permite el desalojo de bloques hacia otras memorias de atracción al objeto de generar espacio libre para el almacenamiento de un nuevo bloque solicitado por el procesador. Una de las principales características de VSR-COMA es que la selección del nodo destino de una operación de reemplazo es independiente del protocolo. Por lo tanto, es posible implementar diferentes mecanismos de selección al objeto de comparar su funcionamiento en la ejecución de programas paralelos.

Para la simulación del protocolo se ha desarrollado un simulador basado en ejecución, denominado EMUCOMA. Este simulador permite la ejecución de aplicaciones paralelas programadas utilizando un modelo de variables compartidas, al objeto de simular el funcionamiento de un sistema multicomputador en el que cada proceso que compone la aplicación paralela se ejecutara en un procesador diferente. Se supone además que todos los nodos de proceso se encuentran unidos a través de un medio de interconexión de tipo bus común. El simulador EMUCOMA se encarga del mantenimiento de la coherencia del espacio compartido de direcciones utilizando el protocolo VSR-COMA, lo que permite que cada una de las aplicaciones acceda a copias actualizadas de los datos.

El objetivo principal del análisis realizado es la comparación del funcionamiento de los diferentes mecanismos de reemplazo existentes con el que se propone para VSR-COMA. Para ello se ha simulado la ejecución de diferentes programas del repertorio Splash-2 en un sistema formado por un conjunto de estaciones de trabajo que utilizara el protocolo VSR-COMA junto con cada uno de los mecanismos de reemplazo estudiados. Los resultados obtenidos se comparan con los que se obtendrían con un sistema COMA que prescindiera del reemplazo, y que por lo tanto dispusiera en cada AM del espacio suficiente para almacenar todo el espacio de direcciones compartido.

El resto del documento está organizado como sigue. La sección II enumera los mecanismos propuestos hasta la fecha para la selección del nodo destino en una operación de desalojo, analizando sus ventajas e inconvenientes. La sección III describe el mecanismo utilizado para la simulación de un sistema débilmente acoplado que utiliza el protocolo VSR-COMA para la gestión de la coherencia. La sección IV resume las características de los programas utilizados para el estudio del rendimiento y los parámetros del modelo analítico del sistema. La sección V muestra los resultados obtenidos en el estudio de los diferentes mecanismos de reemplazo y su influencia en la aceleración. Finalmente, la sección VI recoge las conclusiones de este informe.

II. Mecanismos de reemplazo

Cuando un nodo de un sistema COMA necesita un bloque que no está presente en su memoria de atracción, debe solicitarlo y almacenarlo de forma local antes de poder acceder a él. Puede suceder que

Departamento de Informática, Universidad de Valladolid, Campus Miguel Delibes, 47011 Valladolid (España). Correo electrónico: {diego,benja,agustin}@infor.uva.es.

todos los marcos de bloque del conjunto correspondiente de la memoria de atracción estén siendo ocupados por bloques que el nodo posea en propiedad. En las caches convencionales el problema se resuelve desalojando uno de dichos bloques hacia un nivel de memoria inferior. Sin embargo, en las memorias de atracción no existe dicho nivel, por lo que será necesario enviar el bloque a la memoria de atracción de otro nodo.

El problema del reemplazo en los sistemas COMA tiene dos partes. En primer lugar, debe decidirse qué bloque se desaloja de la memoria de atracción. Este problema admite una solución sencilla, consistente en seleccionar el bloque en función del estado de todos los bloques del conjunto correspondiente [5]. El segundo problema consiste en decidir hacia qué nodo remoto deberá desalojarse el bloque seleccionado. Este problema no tiene una solución única, ya que depende del estado de ocupación de los conjuntos correspondientes en los nodos remotos del sistema, información ésta que el nodo local normalmente no conoce. Se han propuesto varias soluciones diferentes a este problema:

Selección aleatoria: este sistema selecciona de forma aleatoria el nodo destino de una operación de desalojo, enviándole el bloque. Si el nodo destino puede aceptarlo, responde afirmativamente. En caso contrario lo rechaza, por lo que el nodo que ha iniciado el desalojo deberá elegir otro nodo destino. Este mecanismo es el utilizado en el protocolo COMA-F [6].

Selección por consulta: este sistema funciona como sigue. El nodo que inicia el desalojo solicita a todos los nodos remotos del sistema que indiquen su disponibilidad para aceptar el bloque que desea desalojarse. Todos los nodos remotos responden devolviendo un vector de prioridad de cuatro niveles. El nodo que ha iniciado el desalojo selecciona el nodo con mayores posibilidades de aceptar el bloque y se lo envía. Este mecanismo es el que se utiliza en BB-COMA [7] y en DICE [8].

Selección por replicación de información de estado: este mecanismo aprovecha la capacidad de difusión de la red de interconexión para mantener en cada nodo la información de estado de las memorias de atracción de los nodos remotos. De esta manera, cuando deba realizarse una operación de desalojo sobre un bloque, el nodo que la inicia conoce el estado del conjunto correspondiente en todas las memorias de atracción remotas, por lo que puede seleccionar el nodo destino de la operación sin necesidad de generar tráfico adicional en la red de interconexión. Este es el mecanismo propuesto en VSR-COMA [2]. Dado que VSR-COMA supone que los nodos no tienen la posibilidad de reservar la red de interconexión hasta que una transacción se complete, puede darse la posibilidad de que un solapamiento de solicitudes provoque el rechazo del bloque en el nodo destino. En dicho caso el nodo que inició la operación de desalojo deberá reiniciar el proceso de selección de destino.

Se ha estudiado el funcionamiento de los tres mecanismos de reemplazo descritos en la ejecución de programas del repertorio Splash-2, obteniéndose diferentes índices de rendimiento. Los índices obtenidos se han utilizado para calcular la aceleración (*speedup*) en la ejecución de cada programa paralelo con diferente número de nodos. La aceleración se ha calculado a través de un modelo analítico que permite utilizar los parámetros devueltos por el simulador EMUCOMA para calcular índices temporales. El simulador y el modelo analítico utilizado se describen en la sección siguiente.

III. SIMULACIÓN DE VSR-COMA

Se ha desarrollado un simulador que permite evaluar el funcionamiento de los controladores de coherencia de un sistema VSR-COMA durante la ejecución de aplicaciones paralelas. Dicho simulador, denominado EMUCOMA, permite ejecutar aplicaciones paralelas desarrolladas haciendo uso de un modelo de programación de variables compartidas. Nuestro mecanismo de simulación captura las referencias a memoria generadas a través de la ejecución de una aplicación paralela y las resuelve a través de la arquitectura simulada. La resolución de las operaciones de memoria solicitadas por la aplicación permite a ésta continuar con la ejecución del programa.

La figura 1 muestra la estructura de procesos del simulador EMUCOMA. Uno de los principales objetivos del sistema propuesto es la reducción del tiempo de simulación. Para ello se desarrolló una arquitectura basada en *threads*, en donde cada uno de los controladores de coherencia es un *thread* que mantiene actualizada su propia información. Los *threads* se comunican con los diferentes procesos que forman la aplicación paralela a través de un mecanismo de memoria compartida regulada por semáforos. Este mecanismo permite que cada uno de los procesos solicite al controlador de coherencia correspondiente la realización de operaciones sobre la memoria compartida, a través del protocolo de memoria de VSR-COMA [2].

Para poder comparar el funcionamiento de diferentes arquitecturas en la ejecución de aplicaciones paralelas se necesita información acerca de cómo se ha desarrollado dicha ejecución: número de instrucciones ejecutadas, número de accesos a memoria y de eventos de protocolo generados por la arquitectura. El sistema de simulación descrito permite obtener dos clases de índices: los referidos a la aplicación y los de la arquitectura COMA subyacente. Entre los índices más importantes que proporcionan se encuentran los siguientes:

Instrucciones (i): indica el número total de instrucciones ejecutadas por la aplicación. Estos índices se obtienen instrumentalizando el código en ensamblador de la aplicación, de forma que tras la ejecución de cada instrucción se invoca

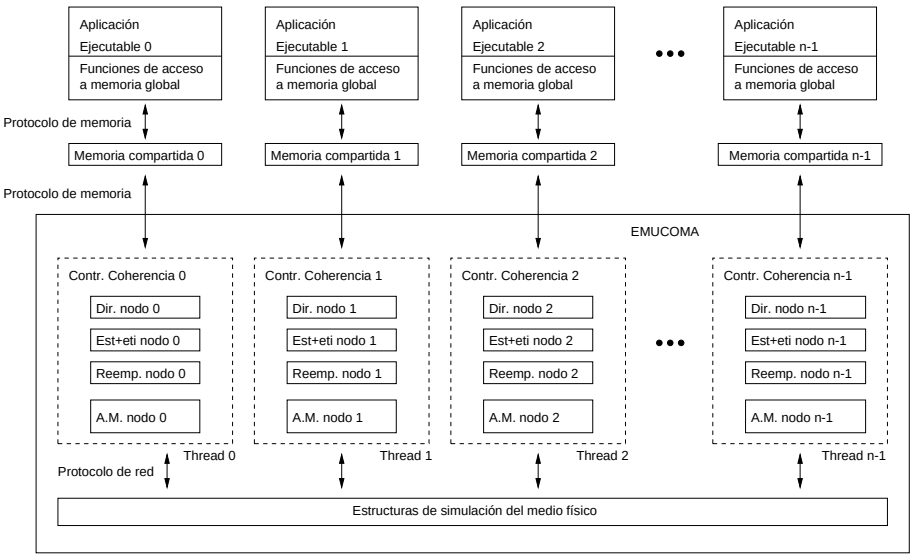


Fig. 1. Estructura de procesos del simulador EMUCOMA.

a una función que incrementa un contador. Dicha instrumentalización de la aplicación se realiza antes del enlace, por lo que el recuento de instrucciones no incluye las instrucciones de las funciones de comunicaciones con el controlador de coherencia ni las de sincronización, sino sólo las referidas a la aplicación.

Accesos a memoria (m): se almacena el número de accesos a memoria para lectura (m_r) y escritura (m_w), tanto a la memoria compartida como a la memoria local de cada proceso.

Accesos a memoria compartida (s): se almacena el número de accesos a la memoria compartida para lectura (s_r) y escritura (s_w). A estos efectos, los accesos para sincronización (a través de operaciones de tipo *Test & Set* y *Fetch & Increment*, suministradas por el protocolo de memoria) se consideran accesos en escritura.

Se ha desarrollado un sencillo modelo analítico que permite obtener resultados sobre el rendimiento en la ejecución de aplicaciones paralelas sobre un sistema VSR-COMA, partiendo de los resultados devueltos por el simulador EMUCOMA. Para construir nuestro modelo analítico hemos considerado un sistema VSR-COMA en el que los nodos son estaciones de trabajo físicamente idénticas, unidas por una red de bus común. En dicho modelo analítico existen unos datos de entrada y unos datos de salida. Los datos de entrada se pueden dividir en dos subconjuntos:

1. Resultados obtenidos en la simulación multiprocesador a través de los índices de rendimiento proporcionados por el simulador y descritos más arriba.
2. Parámetros que describen las características físicas del sistema VSR-COMA concreto. Dentro del conjunto de estos últimos podemos distinguir a su vez dos grupos:
   (a) Parámetros que describen las características físicas del procesador, que son la duración del ciclo de reloj del procesador (τ) y el número de ciclos necesario por término medio para completar cada instrucción.
   (b) Parámetros que describen el comportamiento de la red de interconexión de las estaciones de trabajo, los cuales han sido escogidos representando el comportamiento del sistema de interconexión por medio del modelo LogP [9].

El modelo analítico permite obtener como salida el tiempo que tardaría en ejecutarse la aplicación paralela sobre un conjunto de estaciones de trabajo con un número de procesadores igual al considerado en la simulación y con las características físicas establecidas en los parámetros antes mencionados. A partir de este resultado es inmediato obtener la aceleración para distinto número de procesadores.

Como las distintas estaciones de trabajo pueden ejecutar partes de la carga que no sean exactamente del mismo tamaño, se considera como tiempo de ejecución $T(n)$ en un conjunto de n estaciones de trabajo al mayor de los tiempos calculados para las diversas estaciones de trabajo (nótese que los índices devueltos por el simulador se refieren a cada una de los procesadores involucrados).

El modelo analítico obtiene el tiempo total de ejecución en un conjunto de n estaciones de trabajo como la suma de dos términos:

$$T(n) = t_c + t_r \tag{1}$$

En dicha expresión t_c representa el tiempo consumido para ejecutar toda la parte de carga de trabajo que no hace referencia a la comunicación a través de la red. Frente a ello, t_r representa el tiempo involucrado en las comunicaciones a través de la red necesarias para ejecutar esa carga de trabajo.

Para calcular t_c se considera que el tiempo consumido en la ejecución de cada instrucción se compone de tres partes:

1. Un tiempo fijo para toda instrucción, al que

denominamos t_i.

2. Un tiempo extra fijo t_m si se trata de una instrucción de lectura o de escritura en la memoria.
3. Otro tiempo extra fijo t_s para el caso de que se trate de una instrucción de lectura o escritura en la zona de memoria compartida.

Con ello tenemos que:

$$t_c = it_i + mt_m + st_s = (ic_i + mc_m + sc_s)\tau \qquad (2)$$

Esos tres términos c_i, c_m y c_s representan los tiempos t_i, t_m y t_s expresados en número de ciclos de reloj del procesador. Constituyen los tres parámetros de entrada del modelo que describen el número de ciclos necesarios para completar cada instrucción, y que deben ser ajustados (al igual que τ) en función de la arquitectura física de las estaciones de trabajo utilizadas.

Para calcular t_r, se considera que ese tiempo va a incluir todos los tiempos necesarios para completar, utilizando la red, todas las instrucciones de lectura o escritura que se hayan realizado sobre el espacio compartido y que hayan dado lugar a una falta, esto es, una referencia a memoria que no haya podido ser completada sin acceder a la red. Obtenemos t_r simplemente como

$$t_r = ew \qquad (3)$$

donde w representa el tiempo promedio involucrado en la transmisión por la red de un evento o mensaje.

Para calcular w utilizamos una representación de la red basada en un modelo LogP. En concreto se ha supuesto que la frecuencia de mensajes es lo suficientemente pequeña como para poder despreciar el término de tiempo entre mensajes sucesivos enviados a través de la red ("gap") [10], con lo que w se puede expresar como:

$$w = o_s(l_m) + o_r(l_m) + L(l_m) = a + bl_m \qquad (4)$$

siendo $o_s(l)$, $o_r(l)$ y $L(l)$ respectivamente el coste fijo de envío, el coste fijo de recepción y la latencia de la red de un mensaje de longitud l bytes. Los términos a y b son los parámetros de entrada del modelo que describen el comportamiento del mecanismo de interconexión. El término l_m representa la longitud media de los mensajes intercambiados.

IV. Parámetros de simulación y experimentos realizados

Para las simulaciones se han utilizado diferentes programas del repertorio Splash-2 [11]. Se han escogido un total de seis programas: tres *kernels* (FFT, LU y Radix) y tres aplicaciones (Ocean, Barnes y Radiosity). Cada uno de estos programas ha sido convenientemente adaptado para su uso con Emucoma de acuerdo al modelo de programación de variables compartidas [2].

La tabla I muestra un resumen de las características de los programas Splash-2 utilizados. Además de la información referente al tamaño del espacio de direcciones compartido se indica el número aproximado de instrucciones de cada programa cuando se ejecuta en un único nodo.

Los resultados obtenidos se basan en los índices de rendimiento devueltos tras cada ejecución por el simulador Emucoma. Estos índices se han aplicado al modelo descrito en la sección anterior, utilizando algunos parámetros adicionales que describen las características de la arquitectura de cada estación de trabajo y de la red de interconexión. En concreto, se ha supuesto la utilización como nodos de proceso de estaciones de trabajo de tipo RISC a 167 MHz, lo que supone un valor para τ de 7 ns. De acuerdo con los datos obtenidos de arquitecturas estándar se han considerado los siguientes valores para el resto de parámetros de descripción del sistema: un tiempo t_i fijo para cada instrucción de 1 ciclo (7 ns), un tiempo extra fijo t_m de acceso a memoria de 2 ciclos (14 ns), y un tiempo extra fijo t_s de acceso a la memoria compartida de 10 ciclos (140 ns). En cuanto al medio de interconexión, para el presente estudio hemos supuesto la utilización de una red tipo Myrinet utilizando "fast messages" [12], [13], lo que supone unos valores para los parámetros de la ecuación 4 de $a = 10\mu s$ y $b = 0,026\mu s$.

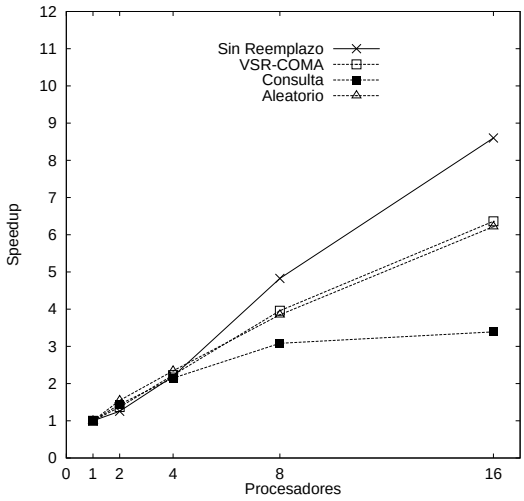
V. Resultados

Se ha realizado un estudio exhaustivo de la influencia de los mecanismos de reemplazo en la ejecución de aplicaciones paralelas. Para ello se ha calculado la aceleración que se obtiene en la ejecución de las aplicaciones anteriormente mencionadas, utilizando el emulador Emucoma y el modelo analítico descrito más arriba.

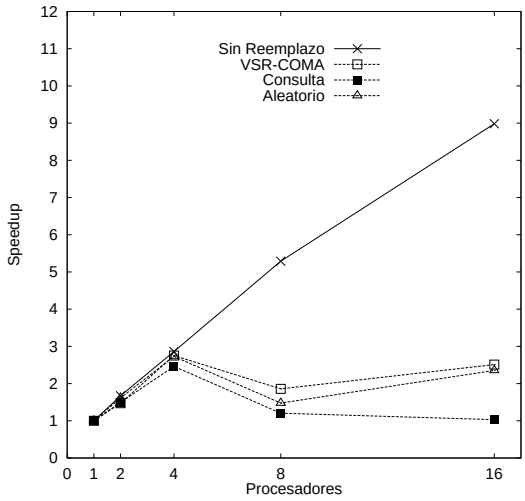
Los resultados obtenidos a través de la utilización de los tres mecanismos descritos se comparan con los que se obtienen utilizando un sistema VSR-COMA con espacio suficiente en cada memoria de atracción para almacenar el espacio compartido de direcciones de la aplicación, lo que implica una ausencia de mecanismos de reemplazo. Este sistema es el utilizado en COMA-BC [14].

La figura 2 muestra las curvas de rendimiento obtenidas para las tres estrategias de reemplazo estudiadas, suponiendo la utilización de bloques de 256 bytes, asociatividad de 4 vías y una presión de memoria del 80%. A esta presión las diferencias en la aceleración obtenida no son muy grandes en términos absolutos, pero el hecho de utilizar el mismo protocolo de coherencia con diferentes estrategias de reemplazo nos permite asegurar que las diferencias que se observan se deben exclusivamente a dichas estrategias.

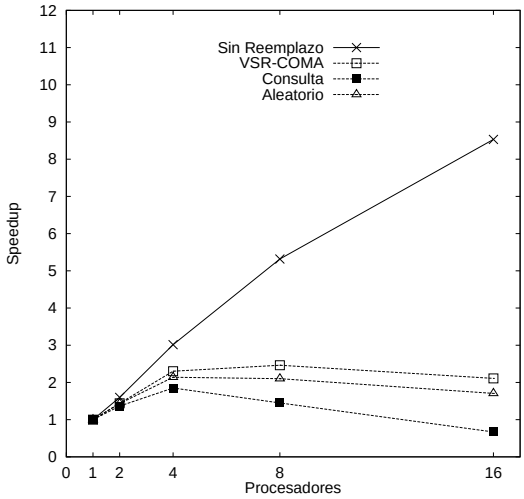
Los resultados obtenidos para el caso de 16 nodos muestra un incremento en la aceleración para la estrategia de VSR-COMA respecto de la estrategia de reemplazo por consulta con factores que van desde el 113% para el caso de Radiosity hasta el 315% pa-



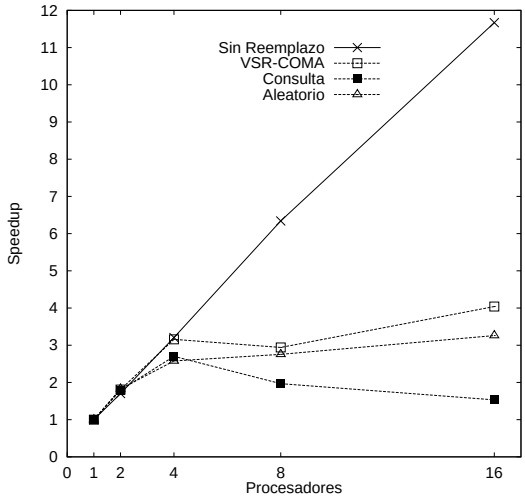
(a) FFT



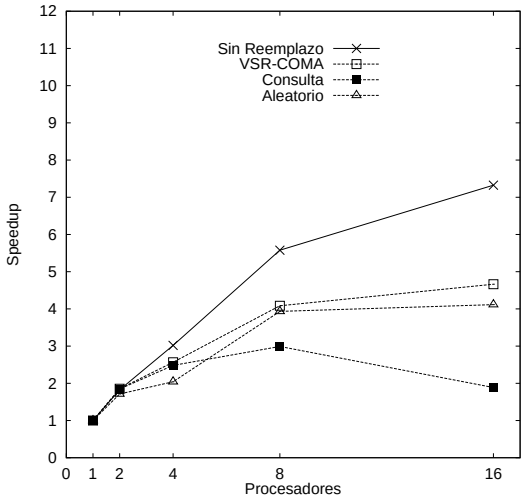
(b) Factorización LU



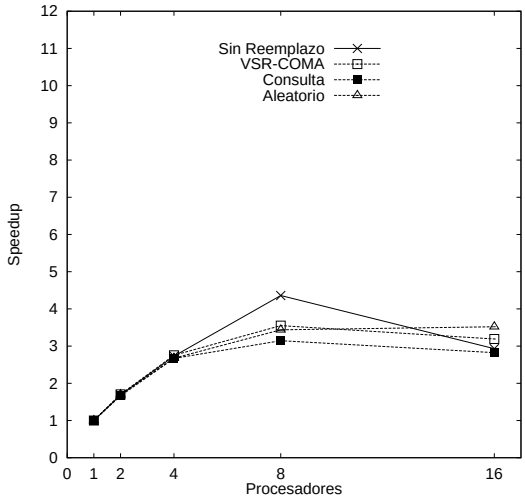
(c) Radix



(d) Barnes



(e) Ocean



(f) Radiosity

Fig. 2. Comparativa de rendimientos para las tres estrategias de reemplazo estudiadas respecto a un sistema sin reemplazo.

Programa	Tamaño del problema	Memoria compartida	Instr. (x10^6)
LU	Matriz de 256 x 256	640 Kb	231
FFT	65.536 puntos	3300 Kb	205
Radix	1.048.576 puntos	9 Mb	663
Ocean	Malla de 258 x 258 km	16 Mb	1.427
Barnes-Hut	4096 partículas	5 Mb	1.802
Radiosity	Modelo "ROOM"	32 Mb	4.090

TABLA I
CARACTERÍSTICAS DE LOS PROGRAMAS SPLASH-2 UTILIZADOS EN LAS SIMULACIONES.

ra Radix. Los restantes programas presentan valores intermedios: 187% para FFT, 243% para LU, 247% para Ocean y 263% para Barnes.

Los incrementos en la aceleración obtenidos por la estrategia de reemplazo utilizada para VSR-COMA en relación con el mecanismo de reemplazo aleatorio son más modestos que para el reemplazo por consulta, y van desde el 102% respecto de FFT hasta el 124% que se obtienen tanto para Radix como para Barnes. Entre ambos valores se sitúan LU (106%) y Ocean (113%). En el caso de Radiosity, se da la circunstancia de que para 16 nodos el mecanismo de reemplazo aleatorio se comporta mejor que el mecanismo propuesto para VSR-COMA, arrojando un incremento con un factor de 90%, es decir, una reducción del 10% respecto a la aceleración obtenida mediante el reemplazo aleatorio.

A la vista de los resultados obtenidos puede apreciarse el bajo rendimiento de la estrategia de reemplazo basada en consulta. Este resultado era previsible si tenemos en cuenta que la selección basada en consulta, pese a seleccionar el nodo destino con mayores posibilidades de acierto, lo consigue a costa de generar un gran número de eventos en la red. Por otra parte, es de destacar el buen comportamiento de la selección aleatoria, que en todos los casos produce mejores resultados que la selección basada en consulta y aceptables en comparación con los producidos por VSR-COMA. En el caso de FFT, los resultados son prácticamente iguales, mientras que para Radiosity llega a superar a VSR-COMA en la ejecución con 16 nodos. En el resto de los casos puede observarse que la aceleración obtenida para la estrategia de reemplazo de VSR-COMA es mayor que la obtenida por el resto de estrategias, lo que demuestra la validez de la estrategia de reemplazo propuesta para dicho protocolo.

VI. CONCLUSIONES

Para comparar el funcionamiento de la estrategia de reemplazo propuesta en VSR-COMA con otras estrategias propuestas en la bibliografía se ha realizado un estudio comparativo a presiones de memoria altas, utilizando el protocolo VSR-COMA como soporte e implementando sobre él las diferentes estrategias de reemplazo. Los resultados muestran que la estrategia de VSR-COMA permite obtener mejores aceleraciones que el resto de estrategias de reemplazo, con incrementos de hasta un 315% para la estrategia basada en consulta y de hasta un 124% respecto de la estrategia de reemplazo aleatorio, uno de los objetivos principales del diseño de VSR-COMA.

REFERENCIAS

[1] Diego R. Llanos Ferraris, Benjamín Sahelices Fernández, and Agustín De Dios Hernández, "Diseño de un protocolo COMA en bus común distribuido con reemplazo," in *Proceedings X Jornadas de Paralelismo, Departamento de Informática, Universidad de Murcia, España*, Septiembre 1999, pp. 241–246.

[2] Diego R. Llanos Ferraris, *VSR-COMA: Un protocolo de coherencia cache con reemplazo para sistemas multicomputadores con gestión de memoria de tipo COMA*, Ph.D. thesis, Departamento de Informática, Universidad de Valladolid, España, Abril 2000.

[3] Fredrik Dahlgren and Josep Torrellas, "Cache-only memory architectures," *IEEE Computer*, pp. 72–79, June 1999.

[4] Erik Hagersten, Anders Landin, and Seif Haridi, "DDM - A Cache-Only Memory Architecture," *IEEE Computer*, pp. 44–54, September 1992.

[5] Farnaz Mounes-Toussi and David J. Lilja, "The effect of using state-based priority information in a shared-memory multiprocessor cache replacement policy," in *Proceeding of the International Conference on Parallel Processing, Minneapolis, MN*, August 1998.

[6] Truman Joe, *COMA-F: A Non-hierarchical Cache Only Memory Architecture*, Ph.D. thesis, Department of Electrical Engineering, Stanford University, 1995.

[7] Anders Landin and Fredrik Dahlgren, "Bus-based COMA - Reducing Traffic in Shared-Bus Multiprocessors," in *Second IEEE Conference on High Performance Computer Architectures*, February 1996.

[8] Sangyeun Cho, Jinseok Kong, and Gyungho Lee, "Coherence and Replacement Protocol of DICE - A Bus Based COMA Multiproccesor," *Journal of Parallel and Distributed Computing*, pp. 14–32, April 1999.

[9] David Culler, Richard Karp, David Patterson, Abhijit Sahay, Klaus ErikSchauser, Eunice Santos, Ramesh Subramonian, and Thorsten von Eicken, "LogP: A Practical Model of Parallel Computation," *Communications of the ACM*, pp. 78–85, November 1996.

[10] K.K. Keeton, T.E. Anderson, and D.A. Patterson, "LogP quantified: The case for low overhead local area networks," in *Proceedings of Hot Interconnects III: A Symposium on High Performance Interconnects*, Stanford, California, 1995, Stanford University.

[11] Steven Cameron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta, "The SPLASH-2 programs: Characterization and metodological considerations," in *Proceedings of the 22nd Anual International Symposium on Computer Architecture*, June 1995, pp. 24–36.

[12] S. Pakin, M. Lauria, and A. Chien, "High Performance Messaging on Workstations: Illinois Fast Messages (FM) for Myrinet," in *Proceedings of Supercomputing'95*, 1995.

[13] Nanette J. Boden, Danny Cohen, Robert E. Felderman, Alan E. Kulawik, Charles L. Seitz, JakovÑ. Seizovic, and Wen-King Su, "Myrinet: A gigabit-per-second local area network," *IEEE Micro*, pp. 29–36, February 1995.

[14] Benjamín Sahelices Fernández, Diego R. Llanos Ferraris, and Agustín De Dios Hernández, "Exploiting parallelism in a network of workstations," *ACM Computer Architecture News*, June 2000.