Una Aproximación a la Verificación del Protocolo de Coherencia Cache del Sistema COMA-BC

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

Resumen—
Se ha llevado a cabo un estudio de verificación del protocolo de coherencia cache COMA-BC. Dicho protocolo se utiliza dentro del sistema COMA-BC, arquitectura multicomputador formada por un conjunto de estaciones de trabajo conectadas a través de una red de tipo bus común, de manera que sobre la memoria distribuida de las estaciones se construye un espacio único de direcciones compartido según el mecanismo de los multiprocesadores COMA. Se ha realizado la verificación por el método de expansión de estados alcanzables de una representación del sistema como máquina de estados finitos para sistemas de 2 y 3 nodos. Para estructuras de este tamaño, ha quedado totalmente descartada la aparición de errores debidos al protocolo. Para tamaños de sistema mayores, el problema de la explosión de estados hace que este método no sea viable. El protocolo plantea serios problemas para que su verificación pueda ser acometida a través de técnicas formales que traten de obviar el problema de la explosión de estados. Se ha realizado un estudio por simulación del funcionamiento del protocolo para sistemas de hasta 100 nodos, no habiéndose detectado ningún error en el protocolo. Además, de los estudios realizados se ha podido deducir el tamaño que debe tener la cola que almacene eventos de red recibidos por cada controlador de coherencia cache, la cual para sistemas de n nodos basta con que tenga un tamaño mayor o igual que n-1 posiciones si se quiere evitar que dicha cola pueda producir algún error de funcionamiento en el sistema a causa de su desbordamiento.

Palabras clave— **Redes de computadores, protocolos de coherencia, COMA, verificación, simulación.**

I. Introducción

COMA-BC [1], [2] es una arquitectura multicomputador. Está constituida por un conjunto de estaciones de trabajo interconectadas a través de una red tipo bus común de intercambio de paquetes. Es un sistema de memoria distribuida virtualmente compartida; esto quiere decir que los distintos procesadores comparten un espacio de direcciones único que es soportado físicamente sobre un conjunto de sistemas de memoria independientes. Esos sistemas de memoria independientes son las memorias locales de cada una de las estaciones de trabajo conectadas en el multicomputador. La gestión de ese espacio de direcciones único se ha diseñado siguiendo el esquema de un multiprocesador tipo COMA [3]. Por ello tiene las siguientes características: 1) Todos lo procesadores comparten un único espacio de direcciones; 2) El espacio de direcciones está dividido en bloques; 3) No existe el concepto de nodo huésped permanente de un bloque: cualquier nodo puede contener o no una copia válida de cualquier bloque; 4) Las copias de los bloques se mueven por la red hacia los nodos que las necesiten en cada momento; 5) Cuando un procesador necesita una copia de un bloque no acude a un cierto lugar que sea el alojamiento permanente del bloque, sino que se la pide a otro nodo que en ese momento posea una copia válida del mismo. De acuerdo con esto, COMA-BC está diseñado según los principios de la máquina de difusión de datos original, en la cual los datos fluyen o se difunden hacia aquellos lugares en los que esos datos se necesitan.

Departamento de Informática. Universidad de Valladolid. E-mail:{agustin,benja,diego}@infor.uva.es

En el sistema COMA-BC cada nodo se comporta como una memoria cache del espacio de direcciones global, y las memorias locales de los nodos son los únicos soportes físicos de la información en memoria. Esas memorias locales reciben el nombre de memorias de atracción. Las memorias de atracción contendrán copias de los bloques del espacio de direcciones común, de manera que en cada instante pueden existir varias copias de un mismo bloque en las memorias de atracción de diferentes nodos. El mantenimiento de la coherencia en este sistema hace necesaria la utilización de un protocolo de coherencia cache, al cual se le conoce como *protocolo de coherencia cache COMA-BC*.

El protocolo de coherencia cache COMA-BC (en adelante nos referiremos a él simplemente como *protocolo COMA-BC*) es un protocolo de escritura invalidante, diseñado específicamente para un sistema de bus común en el que la información circula encapsulada en paquetes que integran mensajes. Es un protocolo híbrido que utiliza técnicas de vigilancia del bus (*snooping*) y un soporte de directorio. En su versión original, en el sistema COMA-BC cada memoria de atracción tiene un tamaño físico capaz de alojar todas las direcciones del espacio compartido; por esa razón no se contempla el reem-

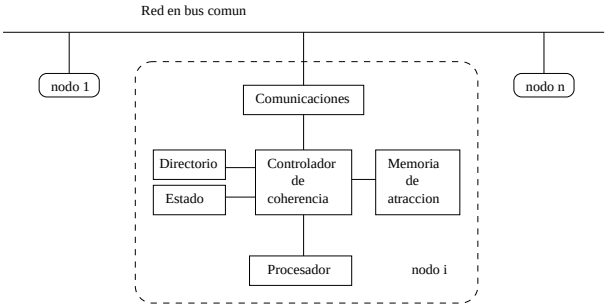

Fig. 1. Arquitectura del sistema COMA-BC

plazamiento de bloques de dicho espacio. De acuerdo con esto, el protocolo COMA-BC no considera operaciones de reemplazamiento de bloques en las memorias cache o memorias de atracción. Una descripción más detallada de este protocolo se puede encontrar en [1], [2]; en la figura 1 aparecen los elementos que conforman un nodo COMA-BC.

El objetivo de nuestro trabajo ha sido llevar a cabo una verificación del funcionamiento de dicho protocolo COMA-BC. Realizar la verificación de un protocolo de coherencia cache consiste en demostrar dos cosas: 1) que nunca se producirá la lectura de un dato erróneo, 2) que nunca se producirá una situación de bloqueo entre los elementos que se ven involucrados en los accesos a memoria (esto es, controladores de memoria y procesadores) que conduzca a la paralización del sistema. Efectuar la verificación del protocolo de coherencia cache implica estudiar todas las posibles situaciones en las que se puede hallar el sistema durante su funcionamiento, de acuerdo con el protocolo, y comprobar que en ninguna de ellas se incurrirá en un error o en un bloqueo definitivo.

Para llevar a cabo la verificación de los protocolos de coherencia, el primer paso es construir una representación del sistema, que funciona con ese protocolo, en forma de máquina de estados finitos; esto es, construir un modelo para ese sistema. Hecho eso, hay que comprobar que en el funcionamiento de ese modelo nunca van a aparecer las condiciones de error indeseadas. Para abordar esta segunda parte se han propuesto diversas técnicas, que se pueden reunir en dos grupos [4]: 1) Técnicas basadas en la expansión de estados alcanzables; 2) Técnicas que recurren a la representación del modelo en base a fórmulas lógicas. Para nuestro estudio hemos utilizado técnicas basadas en expansión de estados alcanzables.

El problema más importante de las técnicas basadas en expansión de los estados alcanzables es lo que se conoce como la *explosión de estados*. Eso significa que al tratar de enumerar todos los posibles estados en los que se puede encontrar el sistema representado como máquina de estados finitos, el número de éstos es tan grande que el problema se hace inmanejable, incluso en los mayores computadores. Las necesidades de memoria y tiempo de ejecución suelen desbordar todas las capacidades de las máquinas conocidas. Es por ello que sólo se pueda realizar con esta técnica, en la práctica, la verificación de los protocolos de coherencia cache para sistemas con un pequeño número de nodos. Existen recientes técnicas que se pueden adaptar a algunos protocolos de coherencia a fin de realizar su verificación para números grandes de procesadores o incluso para un número indeterminado de ellos, pero no siempre es asequible obtener una adecuada representación del modelo del sistema para la aplicación de las mismas.

Por estas razones, la comprobación de la no existencia de errores para sistemas con un gran número de nodos, en general, no puede abordarse de manera exhaustiva, y es necesario recurrir entonces a la simulación del funcionamiento del sistema para poder descartar (con cierta seguridad) la aparición de errores. En nuestro estudio se ha realizado la verificación para sistemas de 2 y 3 procesadores, y se han realizado estudios por simulación para sistemas con mayor número de procesadores.

El resto de este trabajo está organizado en las siguientes partes: 1) Descripción de la técnica de verificación utilizada; 2) Descripción del sistema COMA-BC en términos de máquinas de estados finitos y construcción del modelo; 3) Resultados obtenidos en verificación y simulación; 4) Conclusiones del trabajo realizado.

II. Descripción del la técnica de verificación utilizada

PARA comprobar que un protocolo de coherencia cache no va a dar lugar a errores en el funcionamiento del sistema se puede escoger inicialmente entre realizar una simulación del funcionamiento del sistema ó bien realizar una verificación del protocolo.

Cuando se realiza un estudio por simulación siempre puede subsistir la duda de que alguna determinada situación no haya sido analizada. En una verificación se trata de comprobar exhaustivamente todas las posibles situaciones a las que puede dar lugar el funcionamiento del protocolo, o las que tiene que resolver. Por tanto, lo ideal sería conseguir realizar una verificación del protocolo.

Hoy por hoy, las técnicas aplicadas para la verificación de los protocolos de coherencia cache se pueden clasificar en dos grupos: 1) Aquéllas que utilizan la exploración del espacio de estados alcanzables por el sistema modelado como una máquina

de estados finitos (espacio también denominado gráfico de estados alcanzables, en razón de que todos los estados están conectados por transiciones que permiten ir de unos hacia otros). 2) Aquéllas que realizan el estudio sobre una representación del sistema en forma de un conjunto de fórmulas lógicas, extraídas de dicho sistema modelado como máquina de estados finitos. En éstas últimas se trata de, estudiando dichas fórmulas y sin realizar la expansión del espacio de estados alcanzables, poder deducir la existencia o no de posibles errores en el funcionamiento.

En este trabajo hemos optado por realizar una verificación del protocolo COMA-BC utilizando una técnica de expansión de estados del sistema modelado como máquina de estados finitos. El problema fundamental que plantea una técnica de este tipo es el conocido como *problema de la explosión de estados*. Ese problema consiste en que para comprobar la ausencia de errores es necesario considerar todos los posibles estados en los que puede estar el sistema representado como máquina de estados finitos; y resulta que ese conjunto de estados alcanza un tamaño demasiado grande (inmanejable) salvo en la verificación de sistemas con un pequeño número de nodos. Dentro del conjunto de técnicas que abordan la verificación por expansión de estados se han propuesto diversos procedimientos para evitar el problema de la explosión de estados. En general, esos procedimientos parten de considerar la existencia de ciertas simetrías en el funcionamiento del sistema, las cuales permiten clasificar los estados en clases con representantes canónicos. En primer lugar, si un estado pertenece a una clase y no es erróneo, todos los demás estados de esa clase tampoco serán erróneos. En segundo lugar, en el funcionamiento del sistema, todos los estados sucesores (resultado de dicho funcionamiento) de los estados que están en una clase estarán en las clases en las que están los sucesores del representante canónico de la clase inicial. Por lo tanto, podría realizarse la verificación del protocolo realizando un recorrido a través del gráfico de clases de estados en lugar de a través del gráfico de estados alcanzables. Como el número de nodos del gráfico de clases de estados es mucho menor (en general) que el del gráfico de estados alcanzables, estaría resuelto el problema de la explosión de estados.

Sin embargo, no siempre es fácil (ó posible) obtener la adecuada representación del sistema para las técnicas que tratan de evitar el problema de la explosión de estados, bien a través de la representación en fórmulas lógicas o bien utilizando expansión de clases. En nuestro estudio no hemos abordado la representación en fórmulas lógicas y nos hemos centrado exclusivamente en técnicas basadas en la exploración del espacio de estados; pero sí se ha intentado aplicar alguna técnica de expansión de clases a fin de evitar el problema de la explosión de estados. Al intentar aplicar algunas de estas técnicas hemos detectado tres aspectos del protocolo COMA-BC que hacen muy problemático la aplicación de las mismas y que podemos explicar como sigue. Primero: existe una asimetría esencial en el funcionamiento del sistema. La razón es que cada nodo tiene asignada una identificación que no puede cambiar durante el funcionamiento del sistema; y además esa identificación (que es un número 1,2, ..., n siendo n el número de nodos del sistema) es utilizada por el protocolo cuando cada controlador almacena en su directorio a qué nodo considera él en cada momento como propietario de un determinado bloque. Segundo: cada controlador de coherencia recibe los eventos de red a través de su propia cola FIFO de entrada de eventos. El tamaño que puede alcanzar esa cola FIFO durante el funcionamiento del sistema no puede ser conocido a priori (aunque se le puede establecer un tamaño máximo tal que cuando se intente superar conduzca a una situación de error de funcionamiento). De hecho, el tamaño de una cola FIFO de entrada de eventos podría crecer ilimitadamente. Por ejemplo, considérese un sistema de tres nodos, en el que el nodo 1 y el nodo 2 están escribiendo alternativamente en el mismo bloque. El nodo 3 recibirá por cada operación de escritura un evento RI y otro RRI que no van dirigidos a él, pero que deben ser procesados por él para que actualice su información de directorio al estar cambiando alternativamente quién es el propietario del bloque a causa de las operaciones de escritura de los nodos 1 y 2. Obviamente, los eventos RRI y RI, que reciba en su cola FIFO de entrada de eventos el nodo 3, irán siendo procesados por el controlador de ese nodo; pero podría ocurrir que el controlador 3 procesara dichos eventos a una velocidad inferior a la que son recibidos, y en ese caso el tamaño de su cola FIFO de entrada crecería indefinidamente. Tercero: no es posible hacer una representación del sistema en la que cada cola FIFO aloje como máximo un sólo mensaje. Dicho en otras palabras: el sistema nunca puede funcionar con colas FIFO de entrada cuyo tamaño máximo sea uno. La razón es que cada evento es siempre recibido por todos los controladores. Supongamos que en un sistema con n nodos el nodo **A** emite un evento **a**; todos los nodos lo reciben, con lo que tendrán en su cola FIFO de entrada al menos un evento. Si a continuación un nodo B distinto del A procesa el evento **a** y genera en respuesta a él algún evento **b**, los nodos distintos de A y B tendrán en sus colas los eventos **a** y **b** sin haberlos procesado aún. El problema segundo podría ser aliviado

si se impone alguna condición de funcionamiento al sistema según la cual ningún procesador pueda lanzar nuevas operaciones (READ o WRITE) mientras que algún controlador en el sistema tenga en su cola FIFO de entrada de eventos una ocupación mayor que cierto número de eventos. Pero el problema tercero no puede ser evitado, puesto que habría que limitar el funcionamiento de los controladores, o lo que es lo mismo, habría que imponer la condición que ante un nivel de ocupación 1 de alguna de las FIFO de entrada, los controladores no hicieran nada hasta que el controlador correspondiente hubiera procesado el evento de su FIFO. Se puede comprobar que imponer una limitación de este tipo forzosamente conduce al bloqueo en el funcionamiento del sistema: un sistema COMA-BC puede funcionar con una restricción aplicada sobre el ritmo de envío de operaciones de los procesadores hacia los controladores, pero no con una que evite el procesamiento de eventos a los controladores en determinadas circunstancias. La razón última es que existen eventos que se resuelven forzosamente generando nuevos eventos.

Estos tres problemas nos hicieron desistir de la aplicación de las técnicas de expansión de clases, obligándonos a optar por una técnica de expansión total del gráfico de estados alcanzables, y consecuentemente, limitando el alcance de nuestra verificación a sistemas con un número pequeño de nodos por causa del previsible problema de la explosión de estados. Por esa razón, complementariamente al estudio de verificación, hemos realizado estudios por simulación para sistemas con un número elevado de nodos a fin de aportar datos sobre la fiabilidad del sistema y evaluar algunos aspectos puntuales, como se verá.

Para realizar un estudio de verificación de un sistema representado como máquina de estados finitos pueden utilizarse diversas herramientas automáticas de verificación, entre las que hemos seleccionado Murφ [5]. Murφ es una herramienta que dipone de un lenguaje para describir un modelo de máquina de estados finitos y generar a partir de dicha descripción un verificador (un código ejecutable) que permita realizar ó bien la verificación propiamente dicha (de carácter exhaustivo) ó bien la simulación del funcionamiento del sistema. La descripción del sistema en Murφ se efectúa en un archivo en el que se detallan también qué condiciones deben darse para que una determinada situación deba ser considerada como error en el funcionamiento del sistema representado. Además Murφ permite detectar ciertas condiciones de dead-lock. Existen útiles versiones de Murφ que hacen uso de técnicas formales para evitar el problema de la explosión de estados [6], pero no han sido utilizadas para nuestro estudio por las dificultades antes apuntadas achacables a la propia naturaleza del protocolo COMA-BC. En concreto se ha utilizado Murφ versión 3.0.

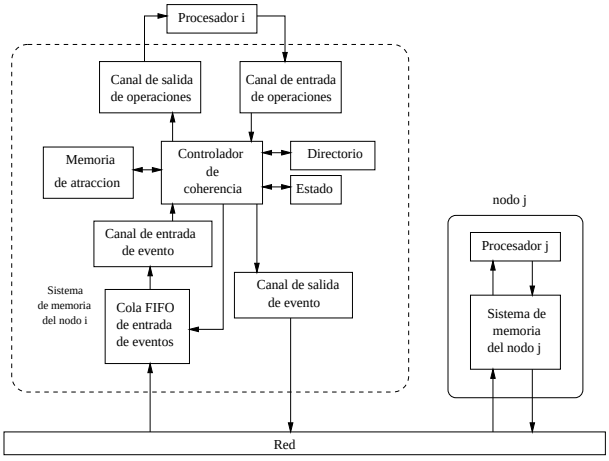

Fig. 2. Elementos del modelo de máquina de estados finitos para el sistema COMA-BC

III. Descripción del sistema COMA-BC representado como máquina de estados finitos

La representación utilizada del sistema COMA-BC como máquina de estados finitos puede verse ilustrada en la figura 2. El sistema está compuesto por una serie de nodos conectados a la red de bus común. A través de la red circulan los eventos. Dentro de cada nodo se distingue el procesador y el sistema de memoria.

El procesador envía y recibe operaciones a y desde el sistema de memoria. Para esa comunicación el sistema de memoria proporciona dos canales, cada uno de los cuales es capaz de almacenar una operación. La información que se almacena en los canales consta de dos campos: 1) Tipo de operación; 2) Datos involucrados en la operación. Cuando un procesador envía una operación a través del canal de entrada de operaciones pasa a un estado de esperando la operación de respuesta a ella; mientras permanezca en ese estado no emitirá nuevas operaciones. El procesador sale de ese estado cuando recibe una operación de respuesta de terminación del acceso iniciado por la operación READ o WRITE, esto es, cuando recibe un Resp-READ o un Resp-WRITE respectivamente, y entonces es cuando da por terminado el acceso de lectura o escritura. Según eso, el procesador puede encontrarse en tres estados, que podemos denominar *libre*, *esperando-lectura* y *esperando-escritura*, de modo que en *libre* puede emitir una operación READ o WRITE, y en los otros no puede emitir ninguna operación, sino que está a la espera de la recepción

de una operación Resp-READ o Resp-WRITE para volver al estado *libre*.

El campo de información referente a los datos involucrados en una operación se refiere al contenido de la información que se mueve. Dado que el mecanismo de verificación considera el funcionamiento del protocolo desde el punto de vista de un bloque cache, las operaciones READ o WRITE del procesador hacen referencia a lectura o escritura sobre un bloque determinado, sin considerar la posición de memoria concreta involucrada en la operación. Y en cuanto a los datos involucrados en las operaciones, en lugar de hacer una representación de todos los posibles datos que se pueden almacenar en una estructura binaria de cierto número de bits, consideraremos únicamente dos tipos de datos: datos que están en estado fresco y datos que están en estado obsoleto [7] . Esto es, un bloque puede contener datos (información) de dos tipos: fresco, obsoleto. Definimos que un dato es fresco si su contenido coincide con lo último que escribió un procesador sobre ese bloque, y decimos que es obsoleto si no es así. Por lo tanto, durante el proceso de verificación del protocolo, en la información de datos referentes a una operación encontraremos *fresco* u *obsoleto*. Cuando un procesador escribe con una operación WRITE, en el campo de datos aparecerá *fresco*; cuando un procesador recibe una operación Resp-READ, en el campo de datos se recibirá la información de datos de ese bloque que contenga la memoria de atracción; esa información será *fresco* u *obsoleto*. Si al procesador llega una operación Resp-READ con datos *obsoleto* quiere decir que el protocolo de coherencia ha dado lugar a la aparición de un error. La información de estado de datos es irrelevante para las operaciones READ y Resp-WRITE, pues esas operaciones no llevan consigo el contenido de información del bloque al que se refieren.

En el canal de entrada de operaciones cabe una sola operación. Ese canal será llenado por el procesador y vaciado por el controlador de coherencia. En el canal de salida de operaciones también cabe una sola operación, siendo éste llenado por el controlador de coherencia y vaciado por el procesador.

El controlador de coherencia realiza acciones en base a las operaciones que recibe del procesador ó de los eventos que recibe desde la red. A su vez, cuando realiza esas acciones genera operaciones que envía al procesador ó eventos que envía hacia la red. Para comunicarse con la red, el controlador de coherencia tiene un canal de entrada de evento y un canal de salida de evento. En un canal de entrada o de salida de evento se almacena la información referente a un evento de red, la cual se compone de los siguientes campos: 1) Codificación del tipo de evento; 2) Número del nodo emisor del evento; 3)Número del nodo al que va dirigido el evento; 4) Estado de la carga útil de datos movida en ese evento, que puede ser *fresco* u *obsoleto*. Este campo sólo es relevante en el caso de los eventos que contienen una carga útil de datos, que concretamente son los eventos RRB y RRI (respectivamente, evento que lleva al nodo receptor una copia de un bloque para su lectura y para su escritura en el caso de producirse un fallo de lectura o escritura). En el caso del evento INV, el campo correspondiente al número del nodo emisor no se utiliza para colocar el número del nodo que envía el evento; en su lugar aparece el número del nodo al que considera como propietario el controlador emisor de INV según su información de directorio.

El controlador de coherencia lee el evento a procesar en el canal de entrada de evento, y una vez procesado, vacía ese canal. Cuando el controlador emite un evento, lo coloca en el canal de salida de evento.

Los eventos llegan al canal de entrada de eventos desde la cola FIFO de entrada de eventos. Cuando el controlador vacía el canal de entrada de eventos, la cola FIFO automáticamente saca el evento de cabeza y lo coloca en el canal de entrada de evento.

Cuando el controlador de coherencia ha llenado el canal de salida de operación ó el canal de salida de evento, se queda a la espera hasta que dichos canales estén vacíos. Si el controlador detecta que la cola FIFO de entrada de eventos está ocupada, no procesará la operación que pudiera aparecer en el canal de entrada de operación hasta que la cola FIFO esté vacía.

El procesamiento que de determinados eventos hace el controlador implica que dichos eventos sean reintroducidos en la cola FIFO de entrada de eventos de ese nodo. Por ello, en el esquema de la figura 2 se representa ello por medio de una flecha que sale del controlador de coherencia para conectar con su cola FIFO de entrada de eventos.

La red toma los eventos que aparecen en los canales de salida de eventos de los controladores y los introduce en las colas de entrada de eventos de todos los nodos, incluida la cola del nodo que ha emitido dicho evento.

En el sistema de memoria de cada nodo, el controlador de coherencia almacena la información de directorio asociada al bloque; esa información es simplemente un número que indica el nodo al que este controlador considera como propietario del bloque. Además almacena también la información que codifica el estado de la copia del bloque (estado de protocolo) en esa memoria de atracción. La memoria de atracción se representa simplemente por medio de una variable que codifica el estado de

la información almacenada en la copia del bloque que está presente en ese nodo, estado que puede ser *fresco* u *obsoleto*. Cuando un procesador finaliza la operación de escritura sobre un bloque, el estado de la copia de ese nodo será *fresco*, y el estado de las copias presentes en los demás nodos será *obsoleto*. Cuando un nodo recibe una copia de un bloque por medio de los eventos RRB o RRI, la información almacenada quedará en estado *fresco* u *obsoleto* según cuál sea el estado de la información que se recibe en el evento.

Con ello quedaría descrito el modelo que representa el sistema COMA-BC como máquina de estados finitos, modelo que ha sido utilizado para llevar a cabo su verificación. Dicho modelo ha sido programado en el lenguaje de descripción de modelos de Murφ.

IV. Resultados obtenidos en verificación y simulación

UTILIZANDO el modelo descrito en el apartado anterior se ha procedido a realizar el estudio de verificación del protocolo COMA-BC utilizando la herramienta de verificación Murφ versión 3.0.

En esta verificación se distinguen las siguientes fuentes de error: 1) Errores por incoherencia entre las distintas copias del mismo bloque: en concreto, la recepción como válido de un dato que ya no representa la última actualización de una información almacenada en algún bloque de la memoria. 2) Errores por recepción, por parte del controlador de coherencia, de algún evento no previsto durante el funcionamiento del sistema. 3) Errores por *deadlock*, entendiendo por tales que se alcance un estado del cual el sistema no puede salir. No se han considerado en este estudio los errores posibles de *live-lock*, esto es, la aparición de secuencias cíclicas de estados del sistema a lo largo de las cuales no se llega nunca a completar ningún acceso a memoria generado desde los procesadores.

Para el sistema COMA-BC de tamaño mínimo, dos procesadores, con el modelo descrito se obtiene un gráfico de expansión de 2010 estados. No se ha detectado ningún error, y la ejecución del verificador ocupó algo menos de 0.1 MB en la memoria principal.

Si se intenta realizar la verificación de un sistema de 3 nodos en el que los procesadores funcionen sin ningún tipo de restricción, salvo las impuestas por el protocolo, la verificación siempre detectaría un error por desbordamiento de alguna de las colas FIFO de entrada de eventos a los controladores. Las colas FIFO de entrada de eventos están diseñadas como *arrays* de longitud fija, de manera que en el momento en que están llenas el verificador genere una señal de error si se intenta introducir algún evento más en la cola. Independientemente de lo grande que fuera el tamaño de las colas FIFO, siempre se podría encontrar en un sistema de tres procesadores un estado en el que alguna FIFO se desbordara, según los problemas que se explicaron en la sección 3. Por tanto, en principio sería imposible realizar la verificación para tres procesadores. Para resolver ese problema podemos establecer una restricción en el funcionamiento de los procesadores, consistente en que ningún procesador pueda generar una nueva operación mientras haya algún controlador al que le quede por procesar algún evento en su cola FIFO de entrada de eventos. Con esta restricción lo que se consigue es ralentizar el funcionamiento de los procesadores, de manera que nunca se acumule un número excesivo de eventos por procesar en ninguna FIFO de entrada de eventos. De este modo evitamos el problema, por ejemplo, de que los nodos 1 y 2 estén alternativamente escribiendo sobre un bloque y que la cola FIFO de entrada del controlador 3 se vaya cargando indefinidamente de eventos RI y RRI si el controlador 3 tardara demasiado en procesar los eventos que entran en su cola FIFO. Estableciendo esta restricción, se consigue completar la verificación para un sistema de tres nodos, sin que se detecte ningún error en el protocolo. En concreto se obtiene un gráfico de estados alcanzados de 1328769 estados, y se ha precisado de una ocupación de memoria de unos 60 MB.

Para 4 nodos, el espacio de memoria que sería necesadio ocupar para la verificación sería superior a 50 GB, lo cual es irresoluble en la totalidad de los computadores convencionales, dejando aparte algunos supercomputadores. Para más de 4 nodos, el problema no prodría resolverse en ningún sistema conocido. Por lo tanto, en la práctica, no es posible realizar la verificación con la técnica expuesta para más de tres nodos. La única forma de poder comprobar, hasta cierto punto, la no existencia de errores generados por el protocolo para sistemas de mayor tamaño es recurrir a estudios de simulación.

El verificador generado por Murφ tiene la posibilidad de ser utilizado para realizar una simulación del funcionamiento del sistema representado, en lugar de realizar una verificación exhaustiva de todos los estados alcanzables. Utilizando esta opción hemos realizado simulaciones del funcionamiento de sistemas con hasta 100 nodos, no habiendo sido detectado ningún error achacable al protocolo.

Una cuestión de interés que se puede plantear sobre el funcionamiento de un sistema COMA-BC es determinar cuál es la capacidad mínima necesaria de las colas FIFO de entrada de eventos tal que éstas nunca puedan desbordarse durante el funcionamiento. En concreto, en verificación se obtiene

que para sistemas de dos nodos la capacidad necesaria es de dos eventos. Para sistemas de tres nodos, operando en simulación y sin aplicar la restricción más arriba propuesta para poder ejecutar la verificación, la capacidad necesaria también es de 2 eventos. Y en general, para sistemas con un número de nodos mayor que tres se obtiene que en simulación no aparece ningún error por desbordamiento de alguna cola FIFO de entrada de eventos si la capacidad de las colas es como mínimo de n-1 eventos siendo n el número de nodos del multicomputador.

V. Conclusiones

DEL estudio realizado se concluye que el protocolo COMA-BC no presenta ningún tipo de error para sistemas de dos o tres nodos, puesto que se ha realizado su verificación, comprobando exhaustivamente todos los estados alcanzables por la representación de un sistema COMA-BC de ese tamaño utilizando un modelo de máquina de estados finitos. Además, para sistemas mayores no se han detectado errores al simular sistemas de hasta 100 nodos.

También se concluye que es imposible realizar una verificación del protocolo COMA-BC para un sistema de más de tres nodos y utilizando una técnica de recorrido exhaustivo del espacio de estados alcanzables en el mencionado modelo. La causa de ello es el problema de la explosión de estados.

Por lo tanto, para tamaños mayores habría que llevar a cabo la verificación utilizando alguna técnica formal que evite el problema de la explosión de estados, bien utilizando una técnica de expansión de clases de estados o bien orientándose hacia la verificación de una representación del sistema construida en términos de fórmulas lógicas. Sin embargo, para conseguirlo habría que realizar una representación del sistema COMA-BC adecuada para la aplicación de tales técnicas. Esta cuestión plantea fuertes problemas, de momento sin resolver, debido a la propia naturaleza del protocolo. Este camino queda abierto y susceptible de ser seguido en un trabajo posterior.

Finalmente, de este estudio se deduce también que el tamaño máximo de las colas FIFO de entrada de eventos, procedentes de la red, para cada nodo basta que sea 2 para sistemas de 2 nodos (con estudios de verificación), y mayor o igual que n-1 para sistemas de n nodos con n mayor que 2 nodos según los resultados obtenidos en el estudio de simulación realizado.

Referencias

[1] B. Sahelices, J.A. Illescas, and L. Alonso, "COMA-BC: A cache only memory architecture multicomputer for non-hierarchical common bus networks," in *6th Euromicro Workshop on Parallel and Distributed Processing*, 1998, pp. 502–508.

[2] B. Sahelices, *COMA-BC: Una arquitectura de memoria sólo cache en bus común no jerárquica*, Ph.D. thesis, Departamento de Informática. Universidad de Valladolid, 1998.

[3] E. Hagersten, A. Landin, and S. Haridi, "DDM: A cache-only memory architecture," *IEEE Computer*, pp. 44–54, 1992.

[4] F. Pong and M. Dubois, "Verification techniques for cache coherence protocols," *ACM Computing Surveys*, vol. 29, no. 1, pp. 82–126, 1997.

[5] D.L. Dill, A.J. Drexler, A.J. Hu, and C. Han Yang, "Protocol verification as a hardware design aid," in *IEEE Int'l Conference on Computer Design*, 1992, pp. 522–525.

[6] C. Norris Ip and D.L. Dill, "Verifying systems with replicated components in Murφ," *Formal Methods in System Design*, 1998, (En prensa).

[7] F. Pong and M. Dubois, "The verification of cache coherence protocols," in *5th. Annual ACM Symposium on Parallel Algorithm and Architecture*, 1993, pp. 11–20.