Reconfiguración de la NoC en la Virtualización de CMPs

Francisco Triviño[1], Francisco J. Alfaro[1], José L. Sánchez[1], José Flich[2] y Santos González[3]

Resumen— **Debido al gran número de nodos que incorporan los actuales sistemas en chip y el escaso grado de escalabilidad que las aplicaciones logran alcanzar, se espera que aumente el número de aplicaciones que se podrán ejecutar de forma concurrente en un mismo sistema. De esta forma, es posible aprovechar gran cantidad de los recursos disponibles. Como consecuencia, se produce un aumento de las interferencias entre las diferentes aplicaciones y por tanto el rendimiento de cada aplicación por separado puede verse seriamente afectado. A nivel de red de interconexión, es posible reducir las interferencias mediante mecanismos de virtualización. Una posible estrategia de virtualización consiste en dividir la red en diferentes particiones tal que cada una puede ejecutar diferentes aplicaciones.**

En este trabajo se propone un mecanismo de reconfiguración de la red para ofrecer soporte de virtualización bajo escenarios realistas. En dichos escenarios, múltiples aplicaciones entran y salen del sistema continuamente. En este caso, el sistema debe proporcionar mecanismos de reasignación dinámica de recursos con el fin de satisfacer las necesidades de las aplicaciones. Los resultados de evaluación muestran un buen entorno de virtualización que permite reducir el tiempo de ejecución de las aplicaciones.

Palabras clave— **Chip Multiprocesador, Redes en Chip, Virtualización, Reconfiguración.**

I. Introducción

Con el fin de aumentar la velocidad de computación, las técnicas actuales de fabricación permiten incluir múltiples nodos de procesamiento en un único chip. Aunque estos nodos no alcanzan la velocidad que proporciona un único y potente procesador de un nodo, varios de ellos mejoran las prestaciones de forma global. Los chip multiprocesador (CMPs) son un excelente ejemplo de estos sistemas [1], [2].

El éxito de los sistemas CMP no sólo depende del número de nodos que incorporan sino también depende de otros recursos tales como el sistema de memoria (caches, memoria principal, protocolo de coherencia, etc.), y el sistema de comunicación. Debido al alto número de componentes a interconectar y para permitir una configuración eficiente entre los recursos, es necesaria una red de interconexión de altas prestaciones. Este es el caso de las redes en chip (Networks on chip, NoCs) capaces de reducir a valores aceptablemente bajos los tiempos de transmisión de la información [4].

Por otra parte, las aplicaciones actuales muestran bajo grado de escalabilidad. Como ejemplo, el estudio realizado en [5] revela el poco grado de escalabilidad obtenido por las aplicaciones PARSEC cuando se consideran todos los componentes involucrados en la arquitectura de un CMP. Del estudio realizado en [5] se puede observar que estas aplicaciones no escalan bien a partir de 16 hilos. Por lo tanto, para aprovechar gran parte de los recursos que ofrecen los CMPs, se espera que varias aplicaciones se ejecuten de manera simultánea. Además, a medida que aumenta el número de nodos, se espera que el número de aplicaciones que se ejecutan de forma simultánea también aumente. Dichas aplicaciones pueden ser de diversa índole (visión por computador, procesamiento multimedia, animación, simulación, etc.) provocando que los patrones de tráfico sean completamente impredecibles.

En este escenario, múltiples aplicaciones comparten todos los recursos que forman el CMP. Como consecuencia, se produce un aumento de las interferencias entre las diferentes aplicaciones. Así, es evidente que si los recursos no se asignan de forma eficiente, el rendimiento de cualquier aplicación puede verse seriamente afectado.

A nivel de red, las interferencias se pueden reducir drásticamente mediante el uso de mecanismos de virtualización. Una red virtualizada consiste en dividir la red en diferentes particiones donde cada partición puede ser utilizada para diferentes aplicaciones y flujos de tráfico. No obstante, la clave de esta propuesta es el hecho de no permitir que el tráfico procedente de una aplicación pueda afectar al de otras aplicaciones. En [6] se ha propuesto un mecanismo de virtualización capaz de reducir los efectos negativos que producen las interferencias. En concreto, el mecanismo se analizó bajo un escenario estático donde cuatro aplicaciones comparten un CMP en el mismo intervalo de tiempo.

No obstante, en un sistema real, las aplicaciones entran y salen del sistema continuamente. En un escenario dinámico, se debe permitir la reasignación de recursos de red a diferentes particiones con el objetivo de adaptarse a las necesidades de las aplicaciones. Por esta razón, en este trabajo, se propone un mecanismo eficiente de reconfiguración de la red para ofrecer soporte de virtualización bajo escenarios realistas, que tiene por objetivo readaptar la NoC para permitir la creación de particiones de forma dinámica.

Este artículo está organizado de la siguiente manera: la sección II muestra el trabajo relacionado. En la sección III se describe, en primer lugar, la propuesta para aislar el tráfico de aplicaciones en una NoC. En segundo lugar, se detalla el mecanismo de reconfiguración de red propuesto. La sección IV presenta la evaluación de prestaciones y los resultados obtenidos. Finalmente, en la sección V se presentan las conclusiones.

[1]Grupo de Redes y Arquitecturas de Altas Prestaciones (RAAP), Universidad de Castilla-La Mancha, e-mail: {ftrivino,falfaro,jsanchez}@dsi.uclm.es.

[2]Grupo de Arquitecturas Paralelas (GAP), Universitat Politècnica de València, e-mail: jflich@disca.upv.es.

[3]Departamento de Informática, Universidad Peruana Cayetano Heredia,santos.gonzalez.t@upch.pe.

II. Trabajo Relacionado

El concepto de virtualización no es nuevo y ha sido aplicado de una forma u otra en los sistemas de la computación desde principios de 1960. Por ejemplo, actualmente múltiples servidores se implementan en máquinas virtuales (VM), que se ejecutan en un único servidor de altas prestaciones. De esta forma, numerosas cargas de trabajo de diferente índole se consolidan en un mismo sistema donde aislar el rendimiento hardware se convierte en una necesidad para la ejecución de aplicaciones con ciertas prioridades, que generalmente vienen marcadas por el usuario o administrador.

En el contexto de los CMPs, el diseño del modelo de virtualización es un proceso en el que intervienen varios elementos. Por ejemplo, la virtualización involucra al sistema operativo y a las aplicaciones que se ejecutan en un mismo sistema. El sistema de memoria debe ser optimizado para minimizar la interferencia entre las distintas máquinas virtuales para aislar mejor la carga de las aplicaciones. Para solucionar este problema, *Michael R. et al.* proponen una variedad de técnicas [7], todas ellas centradas en las jerarquías de memoria que implementan los CMPs. Otro ejemplo puede encontrarse en [8] donde los autores abordan la problemática de compartir los recursos de cache y su utilización para diferentes aplicaciones basándose en parámetros de calidad de servicio. Por desgracia, en todos estos estudios no se tiene en cuenta la red de interconexión.

En cuanto al sistema de interconexión, en [9] los autores introducen el concepto de NoC virtualizada y presentan algunas ventajas basadas en maximizar las prestaciones de la red de interconexión, en mejorar la capacidad de tolerancia a fallos, y en reducir el consumo de energía. Lamentablemente, en este estudio los autores no detallan una metodología de cómo conseguir una NoC virtualizada y, por tanto, no se realiza ningún estudio de evaluación de prestaciones.

En [6], se ha propuesto el uso del mecanismo Logic-Based Distributed Routing (LBDR) [10] como un método eficiente para dividir una NoC en particiones. Concretamente, en este trabajo se analiza una situación estática donde cuatro aplicaciones se ejecuta al mismo tiempo compartiendo un mismo CMP. Esta situación sólo se corresponde con el inicio del sistema donde se asignan el máximo número de aplicaciones posible en función de los recursos disponibles. En una situación real, las aplicaciones entran y salen del sistema continuamente a medida que los recursos son liberados de nuevo. Por tanto, en el estudio [6] no se tuvo en cuenta ningún mecanismo de reconfiguración de la red que permita readaptar las particiones a nuevas aplicaciones.

III. Virtualización Dinámica de una NoC

En esta sección mostramos como se puede aislar el tráfico de diferentes aplicaciones mediante el uso del mecanismo LBDR. También se detalla la metodología de reconfiguración capaz de adaptar el mecanismo de encaminamiento a las necesidades de las aplicaciones.

A. Aislar el Tráfico

Por lo general, un sistema CMP homogéneo está compuesto por nodos. Cada nodo contiene un elemento de proceso, memoria cache de diferentes niveles y el conmutador local que conecta dicho nodo a los nodos vecinos a través de la NoC. Los mensajes generados en el procesador se envían al conmutador a través de una interfaz de red. A continuación, el mensaje se mueve al siguiente conmutador en su camino en función del algoritmo de encaminamiento. El proceso se repite hasta que el mensaje consigue alcanzar su destino. Bajo una situación normal, los enlaces están multiplexados en el tiempo usados por los mensajes pertenecientes a diferentes aplicaciones que se están ejecutando en un mismo instante. Por lo tanto, las aplicaciones compiten por los recursos de red en un entorno caótico donde se producen interferencias a nivel de red. Este hecho reduce considerablemente las prestaciones obtenidas. Por lo tanto, es muy importante aislar el tráfico de diferentes aplicaciones para mejorar las prestaciones.

En [6] se presenta una NoC virtualizada capaz de separar el tráfico generado por diferentes aplicaciones mediante la división de la red en diferentes particiones. En esta situación, los enlaces no están multiplexados entre mensajes pertenecientes a diferentes aplicaciones.

Para conseguir una NoC completamente virtualizada se propuso el uso del mecanismo LBDR [10] que permite la creación de diferentes particiones en una malla 2D con muy pocos recursos hardware. El mecanismo LBDR está formado por dos conjuntos de bits por puerto de salida en cada conmutador. Se utiliza una lógica simple a nivel de bloque que contiene varias puertas lógicas. El primer conjunto consiste en un bit por puerto y permite definir el patrón de conexión de la partición. Cada puerto de salida tiene un bit, *Cx*, que indica si un conmutador está conectado a través del puerto *x*. Por tanto, los bits de conectividad *Cn*, *Ce*, *Cw*, y *Cs* representan la conectividad de un conmutador con los puertos norte, este, oeste y sur, respectivamente. El segundo conjunto consiste en dos bits por puerto y define el conjunto de restricciones de encaminamiento debido al algoritmo de encaminamiento finalmente implementado. Los bits para el puerto de salida este son etiquetados como *Ren* y *Res*. Indican si los mensajes encaminados a través del puerto este pueden tomar la salida por el puerto norte o por el puerto sur en el siguiente con-

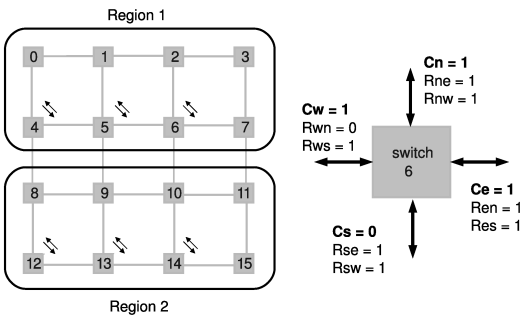
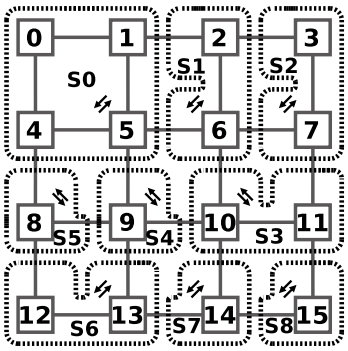


Fig. 1. Ejemplo del mecanismo LBDR.



(a)

Router	Cn	Ce	Cw	Cs	Rne	Rnw	Ren	Res	Rwn	Rws	Rse	Rsw
0	0	1	0	1	0	0	0	1	0	0	1	0
1	0	1	1	1	0	0	0	1	0	1	1	0
2	0	1	1	1	0	0	0	1	0	1	1	0
3	0	0	1	1	0	0	0	1	0	1	0	0
4	1	1	0	1	1	0	0	1	0	0	0	0
5	1	1	1	1	1	1	0	1	1	1	0	1
6	1	1	1	1	1	1	0	1	1	1	0	1
7	1	0	1	1	1	1	0	1	1	1	0	1
8	1	1	0	1	1	0	1	1	0	0	1	0
9	1	1	1	1	1	1	1	1	0	1	1	0
10	1	1	1	1	1	1	1	1	0	1	1	0
11	1	0	1	1	1	1	0	1	0	1	0	0
12	1	1	0	0	1	0	0	0	0	0	0	0
13	1	1	1	0	1	1	0	0	1	0	0	0
14	1	1	1	0	1	1	0	0	1	0	0	0
15	1	0	1	0	1	1	0	0	1	0	0	0

(b)

Fig. 2. (a) Segmentación en zig-zag, y (b) bits LBDR para el algoritmo SR en una malla 2D 4x4 con 9 segmentos.

mutador, respectivamente. En otras palabras, estos bits indican si los mensajes pueden o no cambiar de dirección en el siguiente conmutador. Para el puerto de salida norte los bits son etiquetados como *Rne* y *Rnw*, para el puerto de salida oeste: *Rwn* y *Rws*, y finalmente para el puerto de salida sur: *Rse* y *Rsw*.

La figura 1 muestra un ejemplo del mecanismo LBDR donde un CMP de 16 nodos se ha dividido en dos particiones, cada una de 8 nodos. A modo de ejemplo, en esta figura se detallan los bits de conectividad y de encaminamiento para el conmutador 6. La figura representa con flechas las restricciones de encaminamiento, es decir, el conjunto de dos enlaces consecutivos que no pueden atravesar los mensajes. En este ejemplo se ha aplicado el algoritmo Segment-Based Routing (SR) [11] en cada partición por separado.

Nótese que las rutas de comunicación de cada partición dependen del algoritmo de encaminamiento usado en la red. Dicho algoritmo debe ser lo suficientemente flexible para permitir particiones irregulares y debe ser diseñado teniendo en cuenta los estrictos requisitos aplicados a arquitecturas CMP en cuanto a latencia, consumo de energía y área. El algoritmo SR cumple con dichas restricciones.

B. Mecanismo de Reconfiguración

En esta sección se describe un método efectivo, práctico y rápido para reconfigurar los bits de encaminamiento en una NoC y permitir así la virtualización de los recursos de red (mediante la división de la red en diferentes particiones) en un entorno dinámico.

En primer lugar hay que tener en cuenta que el tamaño y forma de las particiones son elegidas por un gestor de recursos que, por regla general, se ejecuta bajo el sistema operativo. El gestor de recursos puede tener en cuenta diferentes requisitos a la hora de asignar recursos a las aplicaciones tales como: la minimización de la latencia de red entre los elementos de proceso, la posición de los controladores de memoria, la reducción de la fragmentación de red, posibles fallos en la red, ahorro de energía, etc. En nuestro caso, únicamente se tiene en cuenta el número de hilos que componen las aplicaciones, donde un hilo requiere un nodo. A nivel de red, hay que tener en cuenta que el gestor de recursos es independiente del mecanismo de reconfiguración.

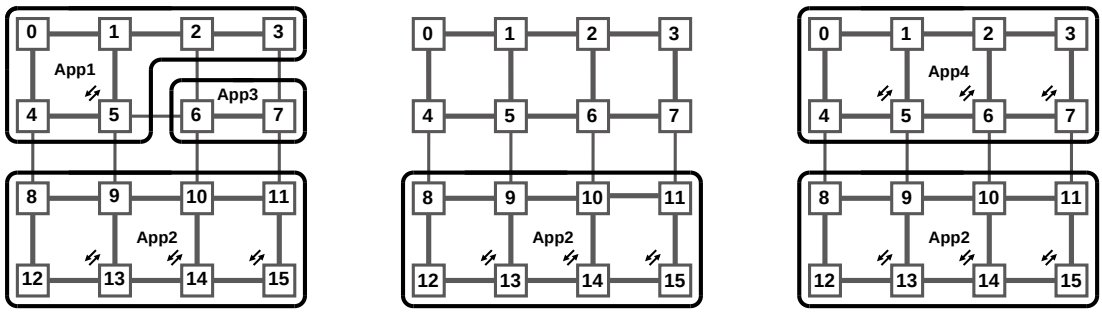
Se parte del hecho de que el algoritmo de encaminamiento es SR [11]. Con el algoritmo SR, es posible pre-configurar un conjunto de bits LBDR para una malla 2D completa y totalmente conectada. Por ejemplo, la figura 2.(a) muestra el resultado de aplicar el algoritmo SR a una malla 4×4[1]. Aunque hay numerosas instancias que se pueden obtener del algoritmo SR, se ha elegido segmentar la red y asignar las restricciones en forma de Zig-Zag de izquierda a derecha empezando de arriba hacia abajo. Este método ha sido analizado obteniendo buenas prestaciones con respecto a otras segmentaciones alternativas [17].

Una vez que se ha obtenido el conjunto de restricciones de encaminamiento (representadas por flechas en la figura 2.(a)), se calculan los bits del LBDR en cada conmutador. Estos bits pueden ser deducidos de forma sencilla teniendo en cuenta la localización de las restricciones de encaminamiento y de conectividad en la red. A modo de ejemplo, la figura 2.(b) muestra los bits para la topología de la figura 2.(a).

Teniendo en cuenta la configuración de encaminamiento anterior, y una vez que el sistema operativo comienza a ejecutar aplicaciones, se necesita identificar las nuevas formas que resultan de la creación de nuevas particiones. Por ejemplo, la figura 3.(a) muestra una situación donde tres aplicaciones han sido asignadas en el CMP donde los bits del mecanismo LBDR se han adaptado consecuentemente. Primero, los bits de conectividad establecen los limites de las particiones (por ejemplo, se configura a 0 el bit de conectividad sur de los conmutadores 2 a 7, mientras que el puerto norte de los conmutadores 6 al 11 se configuran también a 0). Además, las restricciones de encaminamiento se deben configurar para evitar ciclos en las particiones. Dichos bits de encaminamiento se configuran de forma independiente en cada partición. Por ejemplo, el conmutador 5 tiene una restricción bidireccional en las direcciones este-norte y norte-oeste.

Cuando se crea una nueva partición, los bits LBDR se revisan y se actualizan acorde con la forma de la nueva partición. La figura 3.(b) muestra un ejemplo a partir de la situación inicial de la figura 3.(a) donde las aplicaciones App1 y App3 han completado su ejecución. Después, una nueva aplicación (App4) solicita 8 nodos y el sistema operativo le asigna los

[1]No confundir los segmentos SR (líneas punteadas) con las particiones (líneas continuas) del mecanismo de virtualización.



(a) (b) (c)

Fig. 3. (a) Situación inicial, (b) finalizan las aplicaciones App1 y App3 , y (c) comienza la ejecución de la aplicación App4.

nodos de 0 al 7 (Figura 3.(c)). En esta situación, los bits del LBDR para los conmutadores 0 a 7 se deben reconfigurar antes de comenzar con la ejecución de la aplicación App4. Como se puede deducir fácilmente, los bits de conectividad norte de los conmutadores 6 y 7 se deben reconfigurar para permitir la comunicación con todos los nodos de la nueva partición. Lo mismo sucede con los bits de conectividad sur de los conmutadores 2 y 3. Por último, se computan las restricciones de encaminamiento para la nueva partición.

Téngase en cuenta que este proceso de reconfiguración se basa en una reconfiguración estática (no hay tráfico circulando por la red) y afecta sólo a las partes de la red donde no hay mensajes circulando a través de los conmutadores ya que las aplicaciones que los estaban usando han terminado su ejecución. Esto es muy importante porque en otro caso podrían aparecer bloqueos. En ese caso, para evitar los bloqueos, se tendría que detener todo el tráfico de la red antes de reconfigurar la función de encaminamiento, o si se quiere evitar drenar la red previamente, se debería considerar otro mecanismo de reconfiguración más complejo que no afecte el resto de particiones de la red. Gracias al hecho de que no hay interferencias entre el tráfico perteneciente a diferentes particiones, es posible realizar una reconfiguración local sin que afecte al resto de particiones de la red. En el ejemplo anterior, sólo se deben configurar los bits del LBDR de los nodos libres 0 al 7. Por esta razón, el mecanismo de reconfiguración siempre asegura una situación libre de bloqueo.

IV. Evaluación de Prestaciones

En esta sección, se evalúa mediante simulación el entorno de virtualización, que incluye el mecanismo de reconfiguración descrito anteriormente. Para llevar a cabo la evaluación hemos utilizado un entorno de simulación [3] basado en herramientas existentes y orientado a la evaluación de redes en chip. Dicho entorno modela de forma lo suficientemente detallada una NoC, así como los diferentes componentes que forman una arquitectura CMP completa (procesadores, sistema de memoria, sistema operativo, aplicaciones reales, etc.).

El sistema simulado es un CMP homogéneo de 16 nodos. Dicho CMP se estructura en una serie de nodos; cada uno contiene un procesador en orden (UltraSparc III), una cache L1 privada para datos y otra para instrucciones, una cache L2 compartida, un conmutador para comunicarlo con el resto de nodos, unidos con una red de interconexión con topología de malla de dos dimensiones. La coherencia entre los diferentes niveles de cache se preserva mediante el protocolo MOESI. En cuanto al acceso fuera del chip se usa la técnica 3D-Stacking [12] por lo que cada nodo tiene acceso fuera del chip. Para la red de interconexión se asume conmutación wormhole con tamaños de colas de 4 bits. El tamaño de flit definido es de 4 bytes. Por otra parte, se utiliza el mecanismo LBDR que permite la creación de particiones junto con el algoritmo de encaminamiento SR [11]. Además, la red opera a la misma velocidad que los procesadores. Por último, para reducir las interferencias entre la cache L2 de diferentes particiones, se obliga a que los bloques de L2 pertenecientes a una partición sean utilizados por la aplicación que ocupa dicha partición [13]. De esta forma, se consigue aislar el contexto de las aplicaciones a nivel de sistema de memoria.

Como carga de trabajo se han utilizado aplicaciones incluidas en los benchmarks SPLASH-2 [14] y PARSEC v2,1 [15]. La suite SPLASH-2 contiene un conjunto de programas que representan una amplia variedad de aplicaciones científicas y de ingeniería. La suite PARSEC posee una amplia variedad de patrones de computación y comunicación que permiten evaluar las actuales tecnologías de CMP con mayor eficacia.

A. Escenarios

A fin de evaluar el mecanismo de reconfiguración se han considerado diferentes escenarios. En cada escenario se ejecutan 5 conjuntos de aplicaciones diferentes. Cada conjunto de aplicaciones contiene 20 aplicaciones seleccionadas de forma aleatoria de los repositorios de aplicaciones SPLASH-2 y PARSEC v2,1. Los requisitos de las aplicaciones están basados únicamente en el número de hilos. Se ha considerado que cada hilo solicita un nodo diferente. Los requisitos de cada aplicación son elegidos de forma aleatoria desde 2 hasta 8 hilos. El gestor de recursos asigna automáticamente los recursos del CMP a las aplicaciones de forma secuencial. Las aplicaciones son almacenadas en una cola FIFO hasta que el gestor de recursos tiene suficientes recursos para comenzar la ejecución de la siguiente aplicación. Los escenarios se diferencian en el uso que se hace de los recursos del CMP.

El primero consiste en un escenario base (EB) donde cada aplicación utiliza todos los recursos que forman el CMP. En este escenario los hilos que forman cada aplicación son distribuidos de forma aleatoria. En este caso no se considera ningún mecanismo de virtualización, y por tanto, el mecanismo de reconfiguración no es necesario. Así, el tráfico generado por una aplicación en concreto se verá afectado por el tráfico generado por otras aplicaciones.

El segundo caso evaluado (RV, regiones virtuales) parte del escenario base, pero en este caso, la red se divide para crear regiones de forma dinámica. En este escenario, los mensajes generados por diferentes aplicaciones sólo pueden usar los recursos de red pertenecientes a dicha región, por lo tanto, las aplicaciones poseen sus propios recursos de forma dedicada donde el tráfico de una región no puede cruzar otras regiones. Como el tráfico de las aplicaciones debe ser aislado, los recursos deben ser asignados de forma contigua. Por esta razón, se utiliza la estrategia de asignación de recursos presentada en [16]. Hay que tener en cuenta que el gestor de recursos es independiente del mecanismo de reconfiguración. En este escenario cada aplicación es asignada a una región virtual de forma similar que en el ejemplo de la figura 3. Cada partición tendrá diferente tamaño dependiendo de los requisitos de la aplicación a ejecutar (número de hilos).

Por último, hemos considerado un escenario adicional con el objetivo de mostrar el efecto negativo que producen las interferencias en el tráfico de red. Este escenario (DV, dominios virtuales) parte del escenario RV. Sin embargo, los mensajes pertenecientes a un dominio pueden cruzar los límites de otros dominios para alcanzar sus destinos. En este escenario, la carga de red se distribuye a lo largo de todo el CMP dependiendo del algoritmo de encaminamiento usado y suponiendo, en todo momento, caminos mínimos. Como cabe esperar, en el escenario DV no se aplica el mecanismo de reconfiguración de la red puesto que no es necesario. En lugar de eso se ha aplicado el algoritmo de encaminamiento SR en la malla completa tal y como muestra la figura 2.(a).

Por ultimo, cabe destacar que únicamente se aplica el mecanismo de reconfiguración en el escenario RV. En este caso, se ha considerado el tiempo de ejecución adicional cada vez que se ha aplicado el mecanismo de reconfiguración, el cual depende principalmente del tamaño de la partición. Con respecto al gestor de recursos, no se ha tenido en cuenta la posible sobrecarga en tiempo de ejecución de dicha estrategia.

B. Resultados

En esta sección se presentan los resultados obtenidos en el proceso de evaluación. Se han ejecutado 5 conjuntos de 20 aplicaciones en cada escenario. Puesto que en el escenario EB los hilos son asignados a los nodos de forma aleatoria, cada valor obtenido con este escenario es el resultado de treinta simulaciones diferentes, donde el intervalo de confianza se ha establecido al 95 %.

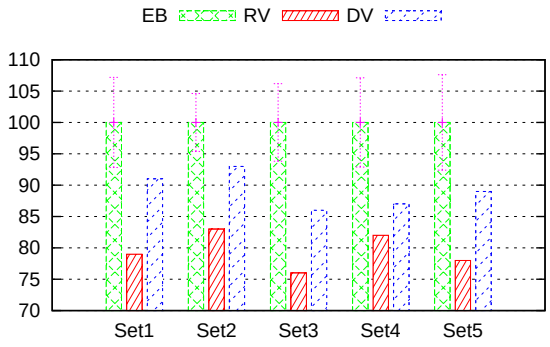


Fig. 4. Tiempo de ejecución normalizado.

La figura 4 muestra el tiempo de ejecución para los diferentes conjuntos de aplicaciones (eje-x). El eje-y representa la diferencia en el tiempo de ejecución total entre los diferentes escenarios (EB, RV y DV). Para ilustrar la variación de rendimiento, los resultados se muestran en términos normalizados comparados con el tiempo de ejecución para el caso EB.

Para el escenario EB, las aplicaciones comparten la totalidad de los recursos del CMP. En este sentido, se obtiene un comportamiento caótico donde las interferencias entre el tráfico de diferentes aplicaciones ocurren constantemente, lo que termina afectando al rendimiento individual de las aplicaciones.

Por otra parte, en el escenario DV se producen menos interferencias de tráfico que en el escenario EB y por tanto las prestaciones son mucho mejores. En concreto, el tiempo de ejecución decrece en un 14 % (para el conjunto Set3) comparado con el caso EB como se puede ver en la figura 4. Si comparamos RV con DV, a pesar de tener la misma asignación de recursos, se obtienen mayores beneficios con RV, puesto que el tiempo de ejecución decrece otro 10 % comparado con el escenario DV (en el conjunto Set3).

Aunque el tiempo de ejecución es la métrica más importante cuando se utilizan aplicaciones reales, en el ámbito de las redes de interconexión son interesantes también otras métricas como la latencia de red y el uso medio de los enlaces. La figura 5 muestra la latencia media producida en la red (a) y la carga media de los enlaces (b) para el conjunto Set1 de aplicaciones, de nuevo en términos normalizados en comparación con el valor obtenido para el caso EB.

En este caso, la latencia obtenida para el escenario EB se incrementa en un 22 % comparada con la del escenario RV, así como un 13 % para el escenario DV. La razón principal por la que el escenario RV obtiene mejores prestaciones se encuentra en el hecho de que el mecanismo de reconfiguración divide la red en diferentes regiones y la distancia media de los mensajes generados por las aplicaciones se reduce de forma significativa. Por tanto, el tráfico de cada aplicación tiene muy baja latencia, lo cual es también una de las razones por las que se obtiene mejores tiempos de ejecución. Cuando los orígenes y destinos de los mensajes no están muy próximos, un mensaje debe atravesar nodos intermedios para alcanzar su destino. Cuando se incrementa el número de saltos, la probabilidad de interferir con otros

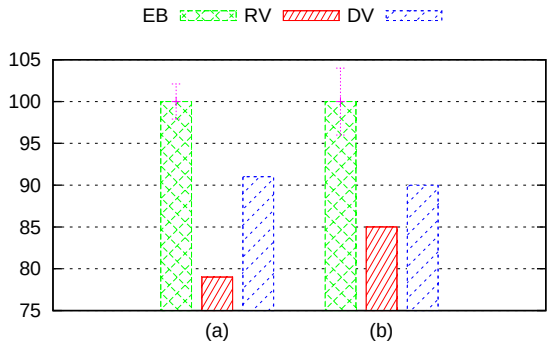


Fig. 5. (a) Latencia media normalizada y (b) uso de enlaces.

mensajes se incrementa también, lo cual se convierte en un aumento de la latencia. Si se comparan los escenarios RV y DV, y teniendo en cuenta que los hilos han sido asignados a los mismos nodos, las diferencias son únicamente debidas a las interferencias entre mensajes.

Por otra parte, el uso medio de los enlaces representa el uso de la red. Esta métrica se ha calculado teniendo en cuenta la carga de todos los enlaces desde el principio de la simulación hasta que todas las aplicaciones han finalizado su ejecución. Como se puede observar, el escenario RV consigue reducir la sobrecarga de comunicación y el tráfico producido puesto que aisla completamente el tráfico de las aplicaciones. Por tanto, se reduce significativamente la carga media de los enlaces cuando se utiliza el mecanismo de virtualización. El escenario RV muestra una reducción del 5 % sobre el escenario DV. Por último, aunque únicamente se muestran resultados para el conjunto Set1 de aplicaciones, la tendencia para la latencia y carga de la red es similar para todos los conjuntos de aplicaciones evaluados.

V. Conclusiones

Este artículo trata de mejorar el rendimiento de las aplicaciones que se ejecutan de forma simultánea mediante el concepto de virtualización. El mecanismo de virtualización permite aislar el tráfico generado por cada aplicación para reducir la sobrecarga de comunicación debida a interferencias entre mensajes pertenecientes a diferentes aplicaciones. En un sistema real, las aplicaciones entran y salen del sistema continuamente. En este caso, la red debe ser capaz de asignar recursos de red a diferentes particiones de forma dinámica. Por esta razón, en este trabajo se describe un mecanismo de reconfiguración para ofrecer soporte de virtualización bajo escenarios realistas donde los recursos del sistema son asignados de forma dinámica.

Se han evaluado tres casos: un modelo base (EB) sin soporte de virtualización y dos modelos basados en virtualización (RV y DV) donde el primero (RV) aisla completamente el tráfico de las diferentes aplicaciones mientras que el segundo (DV) no aisla el tráfico. Se ha observado que el escenario RV puede suponer una importante mejora en las prestaciones que el sistema obtiene. En concreto, se ha observado que el escenario RV reduce (para todos los conjuntos de aplicaciones que hemos simulado) entre un 5 % y un 12 % el tiempo de ejecución comparado con el escenario DV (donde no se aisla el tráfico de aplicaciones). Además, la latencia y carga media de la red siguen la misma tendencia que el tiempo de ejecución. Este hecho se debe a la eliminación de las interferencias de tráfico entre mensajes pertenecientes a diferentes aplicaciones puesto que en ambos casos el resto del CMP (principalmente nodos y cache) se ha particionado de la misma forma.

Agradecimientos

Este trabajo ha sido cofinanciado por el MEC y MICINN del gobierno de España, y por fondos FEDER de la Comisión Europea, con las subvenciones Consolider Ingenio-2010 CSD2006-00046 y TIN2009-14475-C04-03, respectivamente; por la Consejería de Educación y Ciencia de la JCCM con los proyectos PEII11-0229-2343 y POII10-0289-3724, y por el proyecto NaNoC (referencia 248972) que está financiado por la Comisión Europea dentro del programa de investigación FP7.

Referencias

[1] "Tilera Tile-Gx Product Brief," 2010. Available: http://www.tilera.com/pdf/PB025-TILE-Gx-Processor-A-v3.pdf
[2] S. R. Vangal, et al., "An 80-tile sub-100-W TeraFLOPS processor in 65-nm CMOS," *IEEE JSSC*, 2008.
[3] F. Triviño, F. J. Andujar, F. J. Alfaro, J. L. Sánchez, and A. Ros, "Self-Related Traces: An Alternative to Full-System Simulation for NoCs," in HPCS, 2011.
[4] F. Gilabert, F. Silla, M. E. Gomez, M. Lodde, A. Roca, J. Flich, J. Duato, C. Hernández, and S. Rodrigo, *Designing Network On-Chip Architectures in the Nanoscale Era*, J. B. D. Flich, Ed. CRC Press, 2010. Available: http://www.crcpress.com/product/isbn/9781439837108
[5] F. Triviño, J. L. Sánchez, and F. J. Alfaro, "Effect of the CMP Network on the PARSEC v2.1 Benchmark Suite Scalability," INAOCMC, 2010.
[6] F. Triviño, J. L. Sánchez, F. J. Alfaro, and J. Flich, "Virtualizing network-on-chip resources in chip-multiprocessors," *MICPRO*, vol. 35, pp. 230–245, 2011.
[7] M. R. Marty and M. D. Hill, "Virtual hierarchies to support server consolidation," in *ISCA*, 2007.
[8] F. Guo, et al., "From Chaos to QoS: Case Studies in CMP Resource Management," *SIGARCH Comput. Archit. News*, 2007.
[9] J. Flich, J. Duato, T. Sødring, Å. G. Solheim, T. Skeie, O. Lysne, and S. Rodrigo, "On the Potential of NoC Virtualization for Multicore Chips," in *MuCoCoS*, 2008.
[10] J. Flich and J. Duato, "Logic-Based Distributed Routing for NoCs," *IEEE Comput. Archit. Lett.*, 2008.
[11] A. Mejia, J. Flich, and J. Duato, "On the Potentials of Segment-Based Routing for NoCs," in *ICPP*, 2008.
[12] B. Black, et al., "Die Stacking (3D) Microarchitecture," in *MICRO*, 2006.
[13] S. Cho and L. Jin, "Managing Distributed, Shared L2 Caches through OS-Level Page Allocation," in *MICRO*, 2006.
[14] S. C. Woo, et al., "The SPLASH-2 programs: characterization and methodological considerations," in *ISCA*, 2005.
[15] C. Bienia and K. Li, "PARSEC 2.0: A New Benchmark Suite for Chip-Multiprocessors," in *MoBS*, 2009.
[16] A. G. Solheim, O. Lysne, T. Sødring, T. Skeie, and J. A. Libak, "Routing-contained virtualization based on Up*/Down* forwarding," in *HiPC*, 2007.
[17] A. Mejia, "Design and Implementation of Efficient Topology Agnostic Routing Algorithms for Interconnection Networks", PhD dissertation, University of Valencia, 2008.