# Virtualización de una NoC para Chips Multiprocesador*

F. Triviño, JL. Sánchez, FJ. Alfaro
Instituto de Investigación en Informática (I3A)
Universidad de Castilla-La Mancha
{ftrivino, jsanchez, falfaro}@dsi.uclm.es

J. Flich
Grupo de Arquitecturas Paralelas (GAP)
Universidad Politécnica de Valencia
jflich@gap.upv.es

## Resumen

Debido al aumento en el número de cores que se integran en un mismo chip, se espera que también aumente el número de aplicaciones que se podrán ejecutan de forma concurrente en los sistemas en chip. Como consecuencia, el rendimiento de cada aplicación por separado puede verse seriamente afectado debido a interferencias entre todas ellas y en concreto, por el tráfico generado por cada una de ellas.

En este trabajo, se pretende mejorar el rendimiento de las aplicaciones cuando comparten un mismo chip multiprocesador. Se propone aislar el tráfico generado por las diferentes aplicaciones a fin de reducir la sobrecarga de comunicación debida a las interferencias entre mensajes. El modelo se basa en el concepto de virtualización y permite reducir en una cantidad significativa el tiempo de ejecución así como la latencia media.

## 1. Motivación

Con el fin de aumentar la velocidad de computación, las técnicas actuales de fabricación de semiconductores permiten colocar varios nodos de procesamiento en un mismo chip. Los chip multiprocesador (CMPs) son un excelente ejemplo de estos sistemas [7, 11, 2].

Para aprovechar los recursos que ofrecen los CMPs, se espera que varias aplicaciones se ejecuten de manera simultánea. Además, a medida que aumenta el número de cores[1], 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, minería de datos, etc.) provocando que los patrones de tráfico sean completamente impredecibles. Como consecuencia, la carga de trabajo aumenta, lo que puede afectar de manera considerable al rendimiento individual de las aplicaciones.

La figura 1 muestra métricas de rendimiento para una aplicación[2] cuando dispone de todos los recursos (barras rojas) y cuando dichos recursos se comparten por otras aplicaciones cuya carga ha sido modelada mediante tráfico estrés con tasa de inyección de 0,3 (barras verdes). En ambos casos, la aplicación se ejecuta con 16 hilos donde cada hilo se asigna a cada nodo que forma el CMP de 4x4 utilizado en este experimento.

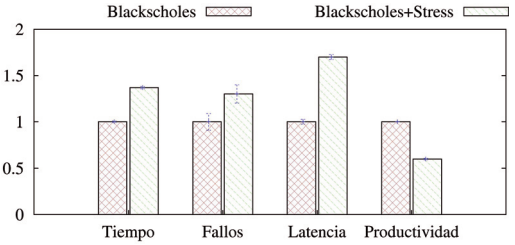



Figura 1: Efecto de tráfico estrés en la aplicación Blackscholes.

Como se puede observar en la figura, el tiempo de ejecución se incrementa en un 38 % cuando se activa el tráfico estrés. No sólo el tiempo

*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 Junta de Comunidades de Castilla–La Mancha con los proyectos PCC08–0078–9856 y POII10-0289-3724, y por el proyecto NaNoC (referencia 248972) que está financiado por la Comisión Europea dentro del programa de investigación FP7.

[1] Usaremos indistintamente el término core o nodo para hacer referencia a los elementos de cómputo de un CMP.

[2] Seleccionada del benchmark Parsec v2,1.

de ejecución se ve afectado sino que el tráfico estrés también tiene un impacto directo en todos los recursos que forman el CMP. Por ejemplo, los fallos en cache se incrementan en 30 % cuando la aplicación comparte los recursos de red.

Por lo tanto, es necesario mejorar el rendimiento individual de cada aplicación. Si los recursos no se asignan de forma eficiente, el rendimiento de cualquier aplicación puede verse seriamente afectado. En este trabajo se resalta este problema y se motiva la necesidad de hacer lo posible para que las aplicaciones alcancen el nivel de prestaciones adecuado.

Nuestra propuesta para conseguirlo se basa en el concepto de virtualización y consiste básicamente en asignar a cada aplicación un subconjunto de recursos que pueda satisfacer sus necesidades de rendimiento. Esto da lugar a agrupar los recursos en varias particiones, donde cada una se compone de un subconjunto de los recursos disponibles en el CMP.

Por supuesto, esta técnica involucra muchos aspectos en la arquitectura de un CMP. Por ejemplo, el sistema operativo junto con un supervisor debe analizar las necesidades de cada aplicación. En base a los recursos disponibles y las necesidades de las aplicaciones el sistema debe decidir si la aplicación se puede o no ejecutar. De ser posible su ejecución, se le deben asignar un conjunto de nodos del CMP en exclusividad o compartir un conjunto previamente creado. Teniendo en cuenta esto último, el sistema debe permitir la reasignación dinámica de los recursos, tales como nodos de procesamiento, bloques de cache, memoria, etc.

En concreto, en este artículo nos centramos en uno de los componentes más importantes de la arquitectura de un CMP: la red de interconexión dentro del chip (NoC). Este componente tiene un gran impacto en las prestaciones. Como veremos después, la latencia media aumenta en gran medida cuando aumenta el número de aplicaciones que se ejecutan al mismo tiempo, influyendo así en el tiempo de ejecución final. Por tanto, minimizar dicha métrica debe ser una prioridad en NoCs.

En este artículo se presenta una alternativa para aislar el tráfico de las diferentes aplicaciones con el fin de reducir los efectos negativos que producen las interferencias entre el tráfico de diferentes aplicaciones. Como veremos, activar el mecanismo de virtualización es mucho más efectivo a la hora de gestionar los recursos.

Este artículo está organizado de la siguiente manera: la sección 2 muestra el estado del arte. En la sección 3 se describen las propuestas para aislar el tráfico de aplicaciones en una NoC. La sección 4 presenta la evaluación de prestaciones mientras que los resultados son llevados a la sección 5. Finalmente, en la sección 6 se presentan las conclusiones.

## 2. Trabajo Relacionado

El concepto de virtualización no es nuevo y ha existido de una forma u otra en la computación y sistemas desde principios de 1960. Por regla general, ofrece una oportunidad para mejorar el rendimiento de las aplicaciones en el contexto de diferentes sistemas. La reciente reaparición de la virtualización se ha aplicado a distintos ámbitos, como las máquinas virtuales, memoria virtual, y los recientes servidores virtuales en los centros de datos. En estos escenarios, múltiples servidores se implementan en máquinas virtuales (VM), que se ejecutan en un único servidor de altas prestaciones. Muchas 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 CMP, 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 [9], todas ellas centradas en las jerarquías de memoria que implementan los CMPs. Otro ejemplo puede encontrarse en [10] 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.

En cuanto al sistema de interconexión, en [5] los autores introducen el concepto de NoC virtualizada y presentan algunas ventajas. Por ejemplo, afirman que, debido al escaso paralelismo que ofrecen las aplicaciones actuales, la virtualización de una NoC proporciona un mecanismo que permite gestionar los recursos de forma que puedan ser asignados a las aplicaciones de una forma eficiente. Otra ventaja se centra en la tolerancia a fallos, ya que un sistema virtualizado permitiría aislar los componentes que presentan fallo, quedando éstos excluidos mediante un esquema de particiones que lógicamente divide la NoC en diferentes regiones. Del mismo modo, en un cierto instante, elementos de proceso y de comunicación que no se estén utilizando pueden ser apagados con el consiguiente ahorro de energía.

En resumen, una NoC virtualizada puede mejorar el rendimiento global del CMP cuando varias aplicaciones se están ejecutando de forma simultánea. Puede ser vista como una red que se divide en diferentes regiones, donde cada una puede encargarse de diferentes aplicaciones y flujos de tráfico evitando las interferencias producidas entre aplicaciones de diferentes regiones. En este sentido, el concepto de virtualización requiere que las rutas seguidas por los mensajes a través de la red puedan ser adaptadas. Como consecuencia, el algoritmo de encaminamiento debe tener en cuenta que los mensajes no puedan cruzar los límites de las regiones. Recientemente, se ha propuesto un método eficiente que permite el particionado de la red mediante el algoritmo de encaminamiento [6]. Dicho mecanismo utiliza una lógica muy simple que permite la creación de regiones en la red de forma eficiente. En concreto, en este artículo se propone el uso del mecanismo LBDR como mecanismo para particionar la red.

## 3. NoC Virtualizada

Por lo general, un sistema CMP homogéneo está compuesto por nodos. Cada nodo contiene un procesador, memoria cache de diferentes niveles y el encaminador o router local que conecta dicho nodo a los nodos vecinos a través de la NoC. Los mensajes generados en el procesador se envían al router a través de una interfaz de red. A continuación, el mensaje se mueve al siguiente router en su camino en función del algoritmo de encaminamiento. El proceso se repite hasta que el mensaje consigue alcanzar su destino.

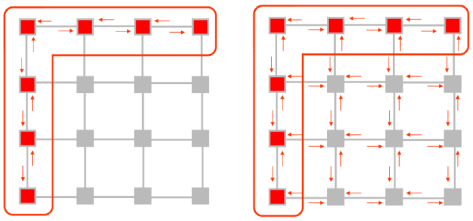
Cuando varias aplicaciones se ejecutan simultáneamente en un mismo CMP, el tráfico generado en la NoC aumenta. Entonces, la latencia de la red crece, y como consecuencia el rendimiento de cada aplicación puede verse seriamente afectado, como en el ejemplo de la figura 1. En este trabajo se propone un mecanismo basado en el concepto de virtualización para la solución de este tipo de problemas.

Cuando se utiliza el mecanismo de virtualización para la creación de varias particiones en un mismo CMP se debe decidir si los recursos de las diferentes particiones pueden o no ser compartidos. En este trabajo abordaremos la virtualización del CMP desde dos enfoques distintos. En el primer enfoque, como resultado del proceso de virtualización los mensajes generados por una aplicación que se ejecuta en una partición determinada sólo pueden usar los recursos de red pertenecientes a esa partición. En el segundo enfoque, por el contrario, también es posible que esos mensajes puedan utilizar los recursos de red de otras particiones. Para distinguir estas dos posibilidades se introducen las siguientes definiciones:

Las **Regiones-Virtuales (RV):** son particiones que se componen de un subconjunto de nodos contiguos y los enlaces que los interconectan. Estos nodos están dedicados exclusivamente a una aplicación o grupo de aplicaciones que comparten estos recursos. En este esquema, el tráfico de una región no puede cruzar a otras regiones.

Los **Dominios-Virtuales (DV):** se componen también de un subconjunto de nodos contiguos pero, en este caso, los mensajes generados por los nodos de una partición pueden utilizar los enlaces (y routers) pertenecientes a otras particiones para alcanzar sus destinos. Por lo tanto, este caso es igual al anterior pero sin la restricción de cruzar regiones.

La figura 2 muestra las diferencias entre regiones virtuales y dominios virtuales. Como se puede ver en (a), los mensajes generados por los nodos de la región (representados por las flechas) sólo pueden utilizar los enlaces y rou-



(a) (b)

Figura 2: (a) Regiones virtuales y (b) Dominios virtuales.

ters que se encuentran dentro de la región. En este caso particular y con altas tasas de transmisión de mensajes, los enlaces se podrían saturar y formar cuellos de botella. Por otra parte, en el caso de dominios virtuales (b), los nodos se pueden comunicar a través de todos los enlaces y routers que forman la NoC y así distribuir la carga. Sin embargo, en este caso el tráfico generado en una partición determinada no está aislado y puede interferir con el resto del tráfico generado en otras particiones. Por el contrario, con regiones virtuales, todo el tráfico está totalmente aislado evitando así las posibles interferencias entre mensajes de diferentes regiones.

Hay que tener en cuenta que las rutas de los mensajes dependen del algoritmo de encaminamiento utilizado en la red. El algoritmo de encaminamiento es la única modificación arquitectónica necesaria para apoyar este modelo de virtualización a nivel de NoC. Este algoritmo debe ser lo suficientemente flexible para permitir la formación de las regiones de forma irregular y debe tener en cuenta las estrictas restricciones aplicadas a los CMPs con respecto a cuestiones tales como la latencia, el consumo de energía y área. Soluciones recientes, como el mecanismo LBDR [6] permite la creación de particiones en una NoC con requisitos hardware muy bajos.

En resumen, nuestra propuesta se basa en el uso efectivo del mecanismo LBDR como un medio para virtualizar la NoC y la creación de las regiones, que dependerá de la asignación de las aplicaciones en el sistema CMP.

## 4. Evaluación de Prestaciones

En esta sección mostramos una breve descripción del entorno de simulación utilizado así como la carga de trabajo utilizada para llevar a cabo las simulaciones. Seguidamente se muestra la configuración base seguida de una discusión detallada de los resultados obtenidos.

### 4.1. Entorno de Simulación

Para llevar a cabo la evaluación de las propuestas presentadas en este artículo, hemos utilizado un entorno de simulación 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.).

Nuestra herramienta de simulación se basa en el sistema Simics-GEMS. Simics [8] es una plataforma capaz de simular gran variedad de procesadores diferentes que modelan funcionalmente todos los elementos hardware del sistema. La herramienta permite ejecutar de manera simulada sistemas operativos y sobre éstos aplicaciones de culaquier tipo, y en particular benchmarks adecuados para la evaluación del sistema. GEMS extiende Simics permitiendo la simulación de arquitecturas multiprocesador. Permite modelar el sistema de memoria utilizado, así como una amplia gama de protocolos de coherencia de caches. Finalmente, para modelar con más precisión el comportamiento de la NoC hemos reemplazado el simulador de red que proporciona GEMS por Noxim [3], un simulador de red más detallado el cual hemos modificado para dar soporte a las propuestas que en este artículo se estudian.

### 4.2. Modelo de CMP

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), una cache L1 privada para datos y otra para instrucciones, una cache L2 compartida, un router para comunicarlo con el resto de nodos, unidos con una red de interconexión con topología de malla de dos dimensiones.

Para la red de interconexión se asume conmutación wormhole con tamaños de colas de 4 flits. El tamaño de flit definido es de 4 bytes. Por otra parte, se utiliza el mecanismo LBDR junto con el algoritmo de encaminamiento SR [1]. Además, la red opera a la misma velocidad que los procesadores.

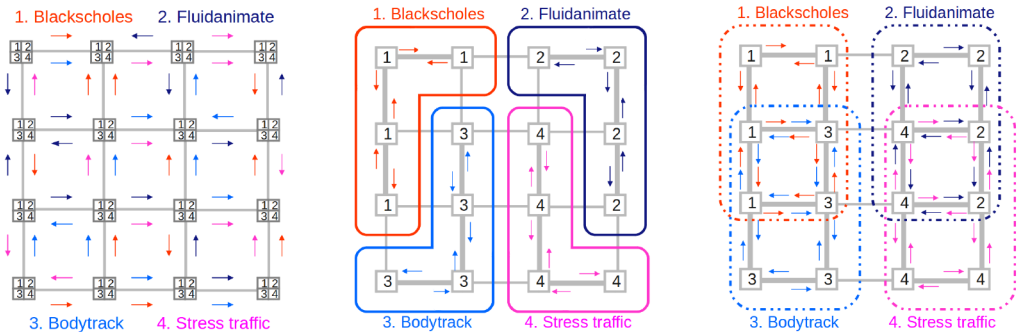



Figura 3: Escenarios: (a) Escenario base (**EB**), (b) Regiones virtuales (**RV**), (c) Dominios virtuales (**VD**)

### 4.3. Carga de Trabajo

Como carga de trabajo se han utilizado aplicaciones incluidas en el benchmark Parsec v2,1 [4]. Esta suite de aplicaciones posee una amplia variedad de patrones de computación y comunicación que permiten evaluar las actuales tecnologías de CMP con mayor eficacia.

Debido a la falta de espacio, sólo se muestran resultados de la aplicación Blacksholes cuando se ejecuta junto con las aplicaciones Fluidanimate y Bodytrack, todas ellas configuradas con input *Simmedium*. Sin embargo, los resultados obtenidos para el resto de aplicaciones y diferentes entradas son similares.

Además de las 3 aplicaciones hemos añadido una carga de tráfico estrés con el fin de observar el efecto de la ejecución de múltiples tareas de forma concurrente. Este tráfico nos permite analizar la tendencia de los resultados cuando se incrementa la carga de red. Es por eso que se han seleccionado tres tasas de inyección diferentes para el tráfico estrés: 0,1, 0,2 y 0,3. Este valor representa una tasa de generación de mensajes constante en los nodos donde el par origen-destino se calcula al azar, así como el tamaño del mensaje (entre 8 y 72 bytes).

### 4.4. Escenarios

A fin de evaluar las propuestas de virtualización se han considerado tres escenarios diferentes. Todos ellos asumen el modelo de CMP base mostrado en la sección 4.2. En cada escenario se ejecutan tres aplicaciones Parsec de forma simultanea además del tráfico estrés. 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. El tráfico generado por una aplicación en concreto se verá afectado por el tráfico generado por otras aplicaciones y por el tráfico estrés. En la figura 3.a se muestra cómo las aplicaciones (con 16 hilos cada una) se asignan en el CMP. Con respecto al tráfico estrés, el destino de los mensajes se calcula al azar teniendo en cuenta todos los nodos del CMP.

El segundo (**RV**, regiones virtuales) parte del escenario base, pero en este caso, la NoC se divide en cuatro regiones como muestra la figura 3.b, donde se asigna cada aplicación a una región diferente. En este escenario, las aplicaciones se asignan con 4 hilos cada una, un hilo por nodo. En este esquema, las aplicaciones poseen sus propios recursos dedicados donde el tráfico de una región no puede cruzar otras regiones. El tráfico estrés se aisla en una región donde los orígenes y destinos de los mensajes sólo pueden ser nodos dentro de su región.

Por último, el tercer 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 tal y como se muestra en la figura 3.c. En este escenario, la carga se distribuye mejor suponiendo, en todo momento, caminos mínimos.

Las geometrías de las regiones en los mecanismos de virtualización se han escogidos como ejemplo para mostrar las posibles interferencias de las distintas aplicaciones corriendo simultáneamente. Estas y otras formas se pueden ob-

tener como resultado de aplicar el mecanismo de virtualización en un escenario más dinámico, donde muchas aplicaciones entran, y salen continuamente del sistema. También se han realizado pruebas con otras geometrías pero se han obtenido resultados similares.

Hay que tener en cuenta que esta comparación puede no ser justa si tenemos en cuenta que las aplicaciones se asignan a los escenarios con diferente número de hilos. De hecho, para el escenario EB podrían beneficiarse de un mayor paralelismo puesto que cada aplicación se ejecuta con 16 hilos en lugar de 4 hilos como ocurre en los escenarios RV y DV. Además, las aplicaciones del benchmark Parsec [4] ofrecen un buen grado de escalabilidad y su ejecución con un mayor número de hilos debería ser mejor, pero las interferencias entre el tráfico de cada aplicación afectará a los resultados como se muestra más adelante.

## 5. Resultados

En esta sección se presentan los resultados obtenidos en el proceso de evaluación. Para todos los casos se consideran escenarios estáticos donde tres aplicaciones junto con el tráfico estrés comienzan su ejecución en un mismo instante. En un escenario más realista, el sistema asigna los recursos de forma dinámica, es decir, las aplicaciones entran y salen del sistema a medida que su ejecución termina. Cuando una aplicación termina su ejecución los recursos que tenía asignados se liberan pudiendo ser asignados a otras aplicaciones. En nuestras pruebas con sólo tres aplicaciones y el tráfico estrés, y en concreto en el caso EB cuando una aplicación finaliza su ejecución los recursos que tenía asociados serán utilizados por las aplicaciones que aún se están ejecutando, por lo que dispondrían de más recursos para su ejecución. Esto sería injusto a la hora de comparar los comportamientos de los diferentes escenarios. Así, con el fin de evitar este efecto, se han recogido estadísticas desde el inicio de la ejecución de las aplicaciones hasta el final de la ejecución de la primera aplicación.

Se ha repetido cada escenario tres veces variando únicamente la tasa de inyección del tráfico estrés. De esta forma se puede evaluar la pérdida de rendimiento debido a la colisión entre mensajes, que es cada vez mayor conforme aumenta la tasa de inyección. Por otra parte, para obtener resultados fiables, cada valor mostrado en las gráficas es el resultado de la media de 30 simulaciones donde se ha variado únicamente la semilla. Además se muestra el intervalo de confianza al 95 %.

Como ya se mencionó previamente, se toman estadísticas desde el inicio de la simulación hasta el momento en que la primera aplicación termina su ejecución, que en este caso es la aplicación Blackscholes, por ser la que menor tiempo necesita. Por tanto, los valores mostrados en cada una de las métricas sólo tienen en cuenta dicha aplicación.

La figura 4 muestra el tiempo de ejecución para los diferentes escenarios (EB, RV y DV) cuando el tráfico estrés varía la tasa de inyección desde 0,1 hasta 0,3. Para ilustrar la variación de rendimiento, los resultados se muestran en términos normalizados comparados con el tiempo de ejecución para el caso EB con 0,1 de tasa de inyección.

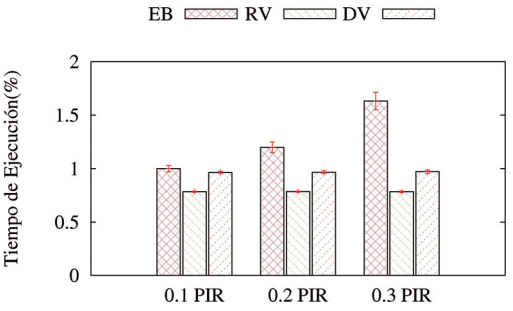



Figura 4: Tiempo de ejecución normalizado.

Para el escenario EB, las aplicaciones junto con el tráfico estrés comparten la totalidad de los recursos del CMP puesto que todas ellas se están ejecutando con 16 hilos, cada hilo en un nodo, compartiendo de este modo toda la red. 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 cada aplicación.

En concreto, el tiempo de ejecución de la aplicación Blackscholes se incrementa en un 73 % aproximadamente cuando el tráfico estrés au-

menta de 0,1 a 0,3 como se puede observar en la figura 4. Como es de esperar, cuando se utiliza la virtualización no se producen interferencias entre los diferentes tráficos y los tiempos de ejecución obtenidos son mejores que para el caso EB (una reducción del 21 % para el tiempo de ejecución en el caso de RV y 3 % para el caso DV, en comparación con el caso EB para 0,1 de tasa de inyección). Para el caso RV no hay variación alguna al incrementar la carga del tráfico estrés puesto que todas las aplicaciones y el tráfico estrés se aislan por completo en regiones virtuales. Aunque el tráfico estrés no afecta para los escenarios RV y DV, hay una pequeña diferencia que hace aumentar el tiempo de ejecución en el escenario DV con respecto a RV. La razón principal se encuentra en el hecho de que, para el caso DV (véase la figura 3.c, el tráfico de la aplicación Blackscholes se ve afectado por el tráfico de la aplicación Bodytrack puesto que sus mensajes pueden cruzar los límites de ambos dominios. Por esta razón, el tiempo de ejecución de la aplicación es mayor que el obtenido en el caso RV, y este valor no varía en función de la tasa de inyección.

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 productividad y latencia de la red. La figura 5 muestra la latencia media producida en la red, de nuevo en términos normalizados en comparación con el valor obtenido para el caso EB con tasa de inyección para el tráfico estrés de 0,1. En este caso, a medida que aumenta la tasa de inyección, la latencia obtenida para el caso EB aumenta considerablemente, pero en los casos de RV y DV permanece constante.

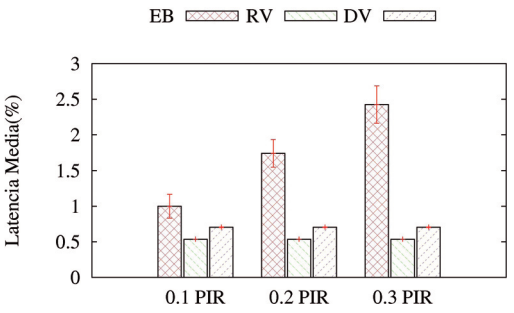



Figura 5: Latencia de red normalizada.

Como se puede observar en la figura 5, la tendencia de esta métrica es la misma que la tendencia mostrada por el tiempo de ejecución. En concreto, se produce un incremento de 142 % al variar de 0,1 a 0,3 de tasa de inyección para el caso EB. En los casos RV y DV se obtienen valores muy bajos en comparación con el caso EB (reducción del 47 % para RV y del 29 % para DV), ya que al dividir la red en diferentes regiones/dominios la distancia media de los mensajes se reduce significativamente. Así, el tráfico de cada aplicación produce muy baja latencia, que es también una de las razones de la obtención de un tiempo de ejecución inferior. Cuando los nodos fuente y destino no están próximos, un mensaje tiene que viajar varios nodos intermedios hasta alcanzar su destino. A pesar de que la distancia no tiene una influencia directa en la latencia utilizando wormhole, cuando el número de saltos que un mensaje necesita para alcanzar su destino crece, la probabilidad de interferir con otros mensajes también crece, lo que termina repercutiendo en la latencia media de los mensajes.

Por último, la figura 6 muestra la productividad media obtenida en la red, de nuevo en términos normalizados en comparación con el valor obtenido para el caso EB con 0,1 de tasa de inyección. Como ya se mencionó, el tiempo de ejecución aumenta considerablemente cuando crece la tasa de inyección del tráfico estrés. Este hecho debería reducir la productividad obtenida en una cantidad significativa. Sin embargo, la tasa de fallos en cache aumenta debido a las interferencias entre mensajes y como consecuencia el número de flits transferidos crece. El efecto combinado de estos dos hechos aparece en la figura 6, donde la reducción en el tiempo de ejecución se compensa por el aumento en la cantidad de flits transmitidos. Como se puede observar, la productividad para el caso EB disminuye ligeramente a medida que aumenta la tasa de inyección del tráfico estrés, siendo esta disminución del 8.7 % al variar de 0,1 a 0,3 la tasa de inyección del tráfico estrés.

Por otra parte, en los casos RV y DV la productividad de la red es menor que en el caso EB (una reducción del 25 % aproximadamente entre los casos RV/DV en comparación con el caso EB) lo que se debe en parte a una menor

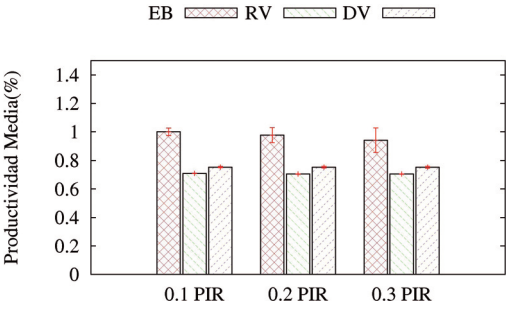



Figura 6: Productividad de red normalizada.

cantidad de flits transmitidos. En cuanto al caso DV se producen interferencias entre las aplicaciones Blackscholes y Bodytrack provocando que la productividad sea ligeramente mayor en comparación con el caso RV (aumento del 5 %).

## 6. Conclusiones

El diseño de un modelo de virtualización es un proceso que involucra varios aspectos de la arquitectura. Cuando los recursos de red en un sistema CMP se comparten por las aplicaciones, las prestaciones para cada aplicación por separado se ven seriamente afectadas. Las interferencias entre mensajes pertenecientes a diferentes aplicaciones producen un efecto negativo que no sólo afecta a las prestaciones de la red de interconexión sino que también afecta a todos los recursos que forman un CMP, como por ejemplo la jerarquía de memoria.

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.

Se han evaluado tres casos: dos modelos basados en virtualización y uno base. Se ha observado que para el esquema base el tiempo de ejecución y la latencia se incrementan a medida que aumenta la carga de la red de interconexión, mientras que la productividad disminuye. Por el contrario, para los casos de virtualización el tiempo de ejecución, latencia y productividad se mantienen constantes cuando el tráfico de red aumenta.

En particular, se ha demostrado que el mejor caso de virtualización reduce en un 21 % el tiempo de ejecución de la aplicación Blackscholes y un 40 % la latencia media de la red en comparación con el escenario base. Este hecho se debe a dos razones principalmente: la eliminación de las interferencias entre mensajes pertenecientes a diferentes aplicaciones y la reducción de la distancia media de los mensajes.

En cuanto a la productividad media, se obtienen valores inferiores para los casos de virtualización. La razón principal radica en la diferencia entre el tiempo de ejecución y la cantidad de flits transmitidos siendo menores para los casos de virtualización. Aunque el tiempo de ejecución se reduce considerablemente, la cantidad de flits transmitidos también se reduce debido a la eliminación de las interferencias entre mensajes de diferentes aplicaciones.

## Referencias

[1] Andres Mejia, et al. On the Potentials of Segment-Based Routing for NoCs. In *ICPP '08*, pages 594–603, Washington, DC, USA.

[2] *TILE64 Processor: A 64-Core SoC with Mesh Interconnect*, 2008.

[3] Noxim Simulator http://noxim.sourceforge.net

[4] Christian Bienia, et al. PARSEC 2.0: A New Benchmark Suite for Chip-Multiprocessors. In *MoBS* '09.

[5] Jose Flich, et al. On the Potential of NoC Virtualization for Multicore Chips. *MuCoCoS'08*, pages 801–807. IEEE, 2008.

[6] José Flich, et al. An Efficient Implementation of Distributed Routing Algorithms for NoCs. *NOCS*, 0:87–96, 2008.

[7] K. Krewell. Best Servers of 2004: Where Multicore is Norm. Microprocessor Report, 2005.

[8] Peter S. Magnusson, et al. Simics: A Full System Simulation Platform. *Computer*, 35(2):50–58, 2002.

[9] Michael R. Marty et al. Virtual hierarchies to support server consolidation. In *ISCA '07*, pages 46–56, New York, NY, USA, 2007. ACM.

[10] Diwaker Gupta, et al. Enforcing performance isolation across virtual machines in Xen. In *Middleware '06*, pages 342–362, New York, NY, USA, 2006. Springer-Verlag New York, Inc.

[11] Sriram Vangal. An 80-Tile Sub-100-W TeraFLOPS Processor in 65-nm CMOS. *l. IEEE Journal of Solid-State Circuits*, 43(1), 2008.