TOPAZ: Un simulador de redes de interconexión para CMPs y supercomputadores

Pablo Abad, Pablo Prieto, Lucía G. Menezo, Adrián Colaso, Valentin Puente, José-Ángel Gregorio

Resumen—**Como en otras áreas de la arquitectura del computador, la investigación en redes de interconexión requiere herramientas de simulación adecuadas, dado que subestimar el modelado de la red puede inducir errores superiores al 50% en la valoración del rendimiento del sistema simulado. Este artículo presenta TOPAZ, una herramienta de software abierto adecuada para el modelado preciso tanto de redes de interconexión de CMPs como de las de los grandes supercomputadores. TOPAZ puede ser empleado de manera independiente con patrones de tráfico sintético y trazas de aplicaciones o formando parte de un sistema de evaluación de sistema completo, como GEMS o GEM5. Para ello, se suministra una avanzada interfaz que facilita el reemplazo del modelo de red de GEMS y GEM5 por TOPAZ. Además, incorpora mecanismos para minimizar el impacto sobre el tiempo de simulación de un modelado detallado de la red. Adicionalmente, TOPAZ ha sido paralelizado para evaluar redes de grandes dimensiones (un millón de *routers*), permitiendo la optimización de los recursos de memoria y el creciente número de *cores* por chip disponibles en los servidores de cálculo actuales.**

Palabras clave—**simulador, redes de interconexión, multiprocesadores en chip, supercomputadores.**

I. Introducción

El campo de las redes de interconexión de sistemas multiprocesadores ha estado restringido a la supercomputación hasta hace pocos años. Sin embargo, con su inclusión dentro del chip [1] la red de interconexión ha pasado a ser un elemento clave en nuevos entornos como los multiprocesadores *on-chip* de propósito general [3][4] o los sistemas on-chip [5]. Diferentes entornos, con requerimientos y restricciones tecnológicas particulares, han diversificado esta área de investigación. Aunque los fundamentos de cada tipo de red son similares, en la práctica, algunos de los parámetros de diseño son factibles en algunos campos, pero no en otros. Por ejemplo, en un contexto on-chip, el ancho de banda disponible es muy elevado y hace factible el empleo de enlaces de comunicación muy “anchos”, mientras que el coste de implementación o el consumo energético del router son parámetros de diseño de primer orden, cosa que no ocurre en las redes off-chip. Muchas otras características, como latencia del enlace, interfaces de red, número de nodos, topología, etc., también pueden tener un impacto diferente dependiendo del entorno.

Esta diversificación es en un obstáculo importante a la hora de diseñar herramientas de simulación de "amplio espectro", incrementándose la complejidad requerida para cubrir los múltiples entornos. Como consecuencia, muchas herramientas de simulación de redes se limitan a un solo campo. Simuladores orientados hacia los CMPs como GARNET [6], integrado como parte de GEMS [7], restringen sus capacidades de simulación a los requerimientos de las redes *on-chip*. Por otro lado, herramientas orientadas a sistemas *off-chip*, como INSEE [8], están diseñadas para la simulación de decenas de miles de nodos. Finalmente, herramientas como NOXIM [9] permiten la simulación de redes heterogéneas, habituales en sistemas on-chip (SOCs).

Dado el mayor esfuerzo computacional de las herramientas más precisas, si no se mantiene un adecuado equilibrio podemos encontrarnos con errores de modelado inadmisibles o tiempos de simulación prohibitivos. Mientras que los simuladores de sistema completo generalmente implementan soluciones a este problema (ajuste dinámico de este equilibrio [10][11] o simulación de diferentes partes del sistema con diferentes niveles de detalle [7]), la mayoría de herramientas de simulación de red no son capaces de adaptarse a los diferentes escenarios.

Este trabajo presenta TOPAZ, una herramienta de simulación de redes de interconexión con un amplio espectro de escenarios de utilización y un adecuado equilibrio entre precisión y velocidad de simulación. Ha sido concebida para ser integrada, con mínimo esfuerzo, en otras herramientas de simulación como GEMS o GEM5, cuya interfaz también se presenta en este documento. El simulador está diseñado para facilitar las tareas de utilización y desarrollo e incluye diferentes mecanismos para minimizar el tiempo de simulación con máxima precisión. Con el fin de facilitar el acceso a la herramienta por otros investigadores y simplificar la adopción de sus propias modificaciones, se ha puesto a disposición pública un repositorio de código fuente y herramientas de gestión de proyectos [12].

El resto del documento está estructurado de la siguiente manera: la sección 2 describe el simulador, la sección 3 explica su integración en un entorno de simulación de sistema completo, la sección 4 describe cómo se puede utilizar el simulador como una herramienta efectiva en redes muy grandes y, finalmente, la sección 5 incluye las principales conclusiones de este trabajo.

II. Descripción del Simulador

TOPAZ es un simulador de redes de interconexión que permite el modelado de una amplia variedad de encaminadores de mensajes, con diferentes *tradeoffs* entre velocidad y precisión. Sus orígenes provienen del simulador SICOSYS [13], concebido para obtener resultados muy próximos, pero a menor coste computacional, a los obtenidos mediante descripciones RTL. Su diseño es orientado a objetos e implementado en C++ mediante aproximadamente 100 clases distribuidas en cerca de 50.000 líneas de código. El simulador soporta la ejecución con *threads* POSIX, su portabilidad es muy alta y se puede ejecutar en cualquier plataforma UNIX con un compilador de C++ estándar.

TOPAZ proporciona descripciones del router con diferentes niveles de detalle. Por un lado, routers *simples* que se describen como un componente único y permite tiempos rápidos de desarrollo y simulación. Por otro

lado, los encaminadores detallados (*complex*) requieren la descripción de cada componente del router por separado, similares a descripciones RTL, alcanzando mucha más precisión a costa de la velocidad de simulación.

Las simulaciones en TOPAZ se puede dividir en tres fases: construcción, ejecución e impresión de resultados. En la primera fase, la red se construye en tiempo de ejecución, de acuerdo a un conjunto de parámetros seleccionados por el usuario. La construcción se realiza de forma jerárquica. El simulador construye la red y los enlaces (topología), la red construye todos los routers y finalmente los routers construyen sus estructuras internas y las interconectan. Cada estructura simulada está asociada a dos clases C++: *components* y *flows*. Los componentes son descriptivos, caracterizando cada estructura y su relación con las restantes estructuras de la red. Por su parte, los flujos establecen cómo se moverá la información dentro de la estructura. Por ejemplo, para una estructura de buffer, el componente determinará su tamaño, el número de puertos o retardo, mientras que el flujo va a determinar su comportamiento de acuerdo con el control de flujo seleccionado.

Durante la fase de ejecución, todos los "componentes" del sistema son visitados de forma iterativa y todos los "flujos" dependientes simulados en cada ciclo. Posteriormente se ejecutan los enlaces (tanto internos como externos al router) que interconectan cada componente. La longitud de la simulación puede ser configurada tanto en términos de ciclos de simulación como en mensajes inyectados. Aunque TOPAZ es una herramienta de simulación conducida por tiempo, algunos flujos pueden ser construidos internamente con máquinas de estados, haciendo estos componentes conducidos por evento.

Junto a las figuras de mérito de la red de interconexión (latencia y *throughput*), la fase de salida puede ser configurada para producir detalles adicionales tales como tiempo de ejecución, resultados por canal virtual, ritmo de inyección y consumo por encaminador, utilización de enlace, cuenta de eventos (útil para la estimaciones de potencia), etc. Si se elige, los resultados también pueden ser suministrados periódicamente durante la fase de ejecución.

A. Estructura

La implementación del simulador se ha orientado hacia una alta portabilidad y una suave curva de aprendizaje. Los ficheros de los directorios de código (/inc y /src) se pueden dividir en las siguientes partes:

constructores SGML (*builder.cpp). Empleados en la fase de construcción y responsables de la interpretación de los ficheros de especificación y de la generación de los objetos requeridos por la simulación. Hay un constructor por cada uno de los tres ficheros de especificación existentes descritos en la sección II.B.

Componentes & Flujos. Representan el hardware que será simulado. Se modelan los componentes internos del router, sus características y retrasos asociados. De esta forma, se establece una relación directa entre una estructura hardware específica y el modelo que nuestro simulador genera.

Patrones de Tráfico (TPZTrafficPattern). Módulo encargado de la inyección de paquetes en la red. Cuando se utiliza el simulador en solitario, se pueden emplear tanto los principales patrones de tráfico (*random*, *transpose matrix*, *bit reversal*, *perfect suffle*, *hot-spot*, etc) como simulaciones basadas en trazas. Trabajando con otras herramientas, para inyectar tráfico generado por una aplicación real se ha definido un patrón "*empty*".

Simulador (TPZSimulator): Este módulo lleva a cabo el proceso de simulación, realizando progresivamente las acciones correspondientes a cada fase de simulación. Se encarga de interpretar todos los parámetros suministrados a través de la línea de comandos, haciendo viable el empleo de simulación basada en secuencia de comandos.

B. Ficheros de especificación

TOPAZ permite experimentar con propuestas arquitectónicas muy diferentes. La especificación del tipo de experimento se lleva a cabo mediante tres ficheros de descripción escritos en SGML y definidos en un fichero adicional llamado "*TPZSimul.ini*". Estos ficheros son interpretados por el simulador en tiempo de ejecución, haciendo innecesaria la recompilación si el usuario desea utilizar alguno de los módulos incluidos en la simulación. A través de la línea de comandos, únicamente necesitamos especificar el nombre de la simulación que queremos correr. Cada fichero especifica los siguientes aspectos:

Simulación (Simulation.sgm): Definición de parámetros generales de simulación, tales como patrón de tráfico y distribución, carga aplicada, longitud del mensaje, etc. La configuración de red que queremos simular es un parámetro adicional en este fichero

Red (Network.sgm): Especificaciones del tipo de red, mediante parámetros tales como topología (malla, toro, etc), dimensiones, retrasos del enlace. El router empleado en cada punto de cruce de la red se define como un parámetro adicional de la red.

Router (Router.sgm): Descripción de los elementos (memoria, conmutadores, multiplexores, etc.) que, interconectados entre ellos, forman el encaminador. También se describen en este fichero los principales parámetros de cada componente del router (retrasos, tamaño, entradas, salidas).

C. Modelos disponibles

La Tabla 1 enumera los modelos suministrados en la versión inicial del simulador. Para la mayoría de los routers existen ambos tipos: simples y detallados. Algunos de los componentes, tales como topologías de bajo grado, tienen como objetivo entornos CMP, mientras que otros son más apropiados para redes off-chip. Aunque algunas combinaciones están fijadas, por ejemplo el Adaptive Bubble Router requiere un control de flujo determinado (*Bubble Flow Control*), en general es posible combinar diferentes topologías, arquitecturas de router y características adicionales. Los modelos suministrados han sido seleccionados para que los investigadores puedan emplearlos inmediatamente, pero también se pretende facilitar el acceso al marco de trabajo para desarrollar nuevos componentes.

D. Implementación Multithread

En general, la paralelización de simuladores de Arquitectura de Computadores es una tarea no trivial y

pocas veces el beneficio compensa el esfuerzo. Si consideramos el hecho de que el número de simulaciones requeridas es muy grande, normalmente es más atractivo el *throughput computing*. No obstante, el tamaño del sistema simulado puede hacer esta solución muy costosa. En particular, las granjas de servidores actuales dedicadas a estas simulaciones emplean CMPs con un número creciente de cores por chip y bajo estas circunstancias, *throughput computing* demanda cantidades masivas de DRAM por servidor lo que incrementa los costes de adquisición. Para maximizar la utilización de estos servidores, TOPAZ ha sido concebido para poder simular redes de supercomputadores con cientos de miles de elementos de computación en paralelo.

TABLA 1. COMPONENTES SUMINISTRADOS CON TOPAZ

Network Topologies				
Ring	Mesh (2D & 3D)	Torus (2D & 3D)	Mdmew (2D)	Sqr Mdmew (2D)
Flow controls				
Virtual Cut Through	Bubble Flow Control	Wormhole	V.C. Flow control	
Multicast Support				
Source-decomposed	Path-based	DOR-Tree based	Adap. (Path-Tree)	
Traffic Patterns				
Random	Bit-Reversal	Perfect-Shuffle	Transpose Matrix	
Tornado	Hot Spot	Local	Trace-Based	
Power consumption (Event count)				
Buffer Write	Buffer Read	VC Allocation	SW Allocation	
SW Traversal	Link Traversal			
Routers				
ID	**Reference**	**Year**	**Level of detail**	
Adaptive Bubble	[14]	2001	complex &simple	
Deterministic Bubble	[15]	1998	complex & simple	
DOR with VC (Dally)	[16][17]	2001	complex & simple	
VCTM (Dally + MC)	[18]	2008	complex & simple	
Rotary Router	[19][20]	2009	complex & simple	
Bufferless Router	[21]	2010	simple	
Bidirectional Router	[22]	2009	simple	
Buffered Crossbar	[23]	1987	complex	
Pipeline Optimized	[24]	2008	complex & simple	
Configuration Parameters				
Buffer Size	Packet Size	Nº of V. Channels	Nº message types	
Router Pipeline	Buffer Delay	Link Delay	Nº physical nets	

Como el motor de simulación es "basado en tiempo", la paralelización de la herramienta se simplifica notablemente. Se parte la red y se asigna cada parte a diferentes *threads* de simulación esclavos. La frontera del grupo de componentes asignado a cada *thread* esclavo debe sincronizarse periódicamente con los *threads* vecinos. Para facilitar la portabilidad, su paralelización está basada en *threads* POSIX.

III. CHIP MULTIPROCESSORS: INTEGRACIÓN CON HERRAMIENTAS DE SIMULACIÓN DE SISTEMA COMPLETO

A. *Integración Topaz-GEMS*

TOPAZ es completamente modular y puede utilizarse de forma abstracta en herramientas de simulación de sistema completo, suministrando la API necesaria para conectar simuladores externos sin excesivas modificaciones de código. El desarrollador deberá crear una interfaz, de acuerdo a las peculiaridades del simulador y las necesidades del sistema. TOPAZ está conectado al subsistema de memoria de Ruby. Como es un componente común en GEM5 [11], FeS2 [25]. y GEMS [7], TOPAZ trabajará con todos ellos sin problemas. Se suministra en la página pública de TOPAZ [12] un parche para la versión 2.1 de GEMS que realiza la integración. También se suministra un clon del repositorio de desarrollo de GEM5.

Con el fin de minimizar los efectos sobre el rendimiento que supone el incremento de precisión, TOPAZ proporciona tres mecanismos diferentes. Primero, su implementación multithread posibilita que las simulaciones de red corran en un thread separado respecto del thread principal, minimizando el overhead introducido por TOPAZ al tiempo de simulación. En segundo lugar, se ha diseñado una interfaz adaptativa entre ambos simuladores, que adapta el nivel de precisión en la simulación al nivel de contención en la red. El nivel de contención se estima a través de la cantidad de paquetes que circulan por la red en un instante preciso. Cuando el número de paquetes en la red está por debajo de un umbral predefinido, se deshabilita TOPAZ y únicamente se simula Ruby. Si se sobrepasa el umbral, se activa TOPAZ. Finalmente, la tercera técnica se basa en la modificación de la complejidad del router. Para casi todos los routers suministrados se pueden emplear dos modelos de diferente precisión, uno simplificado, para las etapas iniciales del diseño y otro modelo, más detallado, para las posteriores.

A. *Efecto de la Precisión en la Simulación de la Red*

El simulador de red original de Ruby es muy simple y únicamente modela la contención a nivel de enlace. Aunque esto acelera la simulación, si la carga aplicada es alta, introduce un error nada despreciable en la estimación de la latencia de red. Versiones recientes de GEMS/GEM5 han introducido la posibilidad de emplear GARNET [6] como un sustituto del simulador original. Aunque mucho más detallado que el original, GARNET tiene una flexibilidad limitada con una arquitectura de router fija y únicamente contempla el cambio de unos pocos parámetros de la red. Como se muestra en la Tabla 1, TOPAZ es mucho más flexible y suministra un conjunto de arquitecturas de encaminador más completo. La interfaz TOPAZ-Ruby se ha desarrollado para soportar cualquier configuración de red CMP o SMP, empleando topologías tanto regulares como definidas por fichero.

Con el fin de demostrar la relevancia que puede tener una simulación precisa de la red, se han llevado a cabo simulaciones de diversas aplicaciones y protocolos de coherencia. Restringiendo el estudio a GEMS, se han considerado veinte cargas de trabajo, incluyendo aplicaciones multiprogramadas y multi-hilo (numéricas y de servidor) ejecutándose sobre el sistema operativo OpenSolaris 10. Las aplicaciones numéricas son la suite completa de las NAS Parallel Benchmarks (versión 3.2 de la implementación OpenMP [26]) y cuatro benchmarks de la suite PARSEC [27].

Los benchmarks de servidor corresponden a la suite completa de Wisconsin Commercial Workload [28], liberada por los autores de GEMS en la versión 2.1. Las restantes corresponden a cargas multiprogramadas empleando parte de la suite SPEC CPU2006 en modo "rate" (donde se reserva un core para ejecutar los servicios del sistema operativo) [29]. Cada aplicación se ejecuta múltiples veces con perturbaciones aleatorias en el tiempo de acceso a memoria con el fin de alcanzar intervalos de confianza del 95%. Este número de aplicaciones permite barrer demandas de red muy distintas con el fin de conocer el margen de error a consecuencia del diferente modelado de la red.

Se ha empleado un CMP de 16 procesadores, con procesadores OoO tipo Nehalem [30] y una red en malla 4×4 conectando los procesadores y los 16 bancos de L2. Analizamos cómo se comporta el sistema con los protocolos de coherencia suministrados: *MOESI_CMP_token* y *MOESI_CMP_directory*. Todos los parámetros de configuración escogidos en la comparación son suministrados en el repositorio de GEMS en [12].

Se ajustan todos los simuladores para tener configuraciones similares de encaminador. En particular, se escoge un encaminador segmentado *wormhole* de 5 ciclos suministrado por GARNET [6]. Se supone 1 ciclo de paso entre routers. En el simulador de red simple de Ruby, se ajusta la latencia para que coincida con la de los routers de GARNET. Se emplean 4 canales virtuales (uno por clase de tráfico) y 10 flits de 16 bytes de buffer por canal virtual. TOPAZ se ha configurado empleando el mismo router en su implementación simple y compleja y haciendo coincidir todos los parámetros de configuración.

B. *Protocolo MOESI_CMP_directory Coherence*

Este protocolo de coherencia se caracteriza por aplicar una carga limitada a la red y por tanto puede tomarse como referencia de que una simulación cuidadosa de red no debería tener mucho impacto sobre la estimación del rendimiento del sistema. La Figura 1.a muestra el tiempo de ejecución normalizado para las cargas de trabajo consideradas. Si comparamos las implementaciones simple y compleja de TOPAZ, las diferencias son pequeñas, pero apreciables con respecto a los otros simuladores. Teniendo en cuenta que la herramienta de la que deriva TOPAZ [13] es capaz de alcanzar un error menor del 3% con respecto a simuladores hardware, parece razonable emplear TOPAZ como punto de referencia. Por un lado, está claro que el modelado de la contención en el simulador original es demasiado optimista, introduciendo un error considerable en la estimación del tiempo de ejecución de cada aplicación. Por otra parte, el modelado de red de GARNET parece demasiado pesimista. Como en el caso del simulador original, con aplicaciones de carga baja el error es pequeño, pero con elevada contención las diferencias en el tiempo de ejecución pueden ser hasta tres veces mayor. En promedio, el error en el tiempo de ejecución es del 25%, optimista en caso del simulador de red original y pesimista en el caso de GARNET. Por lo tanto, en un escenario no muy demandante, el modelado inadecuado de la contención podría inducir notables errores en los resultados de evaluación.

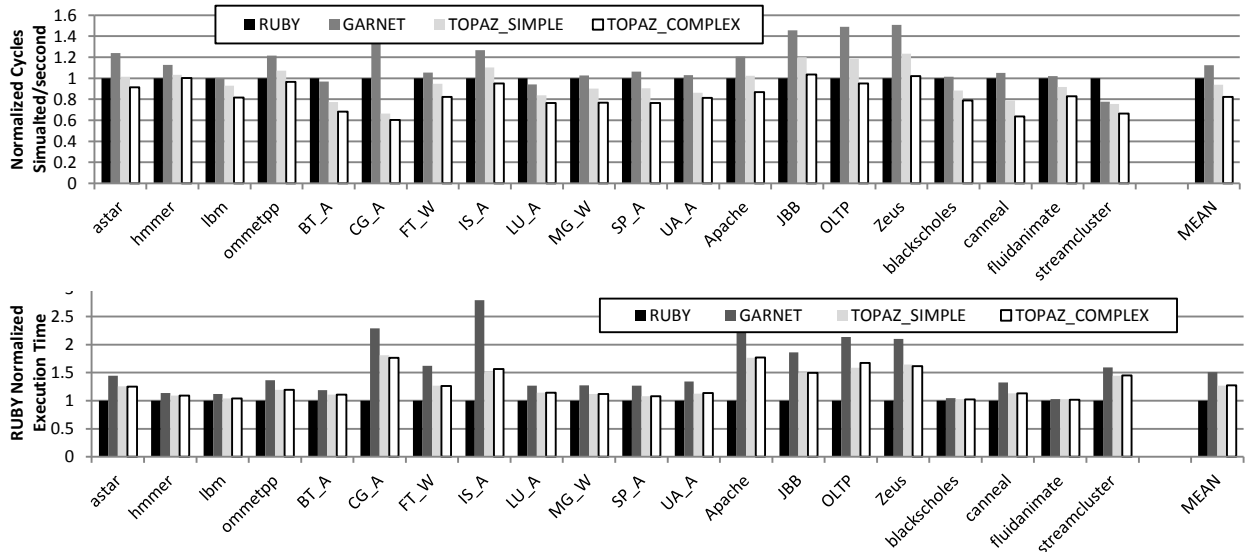

Figura 1. Protocolo de coherencia Directorio. (a) Tiempo de ejecución. (b) Rendimiento de las diferentes herramientas de simulación.

La Figura 1.b muestra el rendimiento de cada simulador en términos del número de ciclos simulados por segundo. Como se puede observar, el router más preciso de TOPAZ (complex) tiene un impacto nada despreciable sobre el rendimiento, incrementando el tiempo de simulación en torno al 20%. El empleo de modelos simples atenúa este enlentecimiento promedio, resultando una aproximación razonable, en torno al 5%. Sorprendentemente, GARNET es el de mayor rendimiento, mejorando en promedio un 10% el simulador GEMS original. Ello se explica porque su modelado pesimista de la contención incrementa la latencia promedio percibida por los procesadores, haciendo decaer la actividad en el resto del sistema e incrementando el número de ciclos simulados por segundo.

C. *Protocolo de coherencia MOESI_CMP_token*

Al contrario que el directorio, este protocolo se caracteriza por unos elevados requerimientos de red debido al tráfico multi-destino. Este ejemplo podría ser considerado como un punto de referencia donde una cuidadosa simulación de la red tiene un gran impacto sobre el rendimiento final del sistema y sobre la

velocidad de simulación. Dado que GARNET no tiene soporte nativo para tráfico multicast, se ha decidido prescindir de este simulador en la comparación.

La figura 2 muestra cómo se comporta un CMP para cada aplicación y cómo los elevados requerimientos de ancho de banda de este protocolo [31] incrementan la contención y el modelado optimista de GEMS induce errores apreciables en los tiempos de ejecución. En algunos casos, como Apache e IS, el error observado es superior al 100%, mucho mayor que el que aparecía en el caso del directorio. Como se ha mencionado anteriormente, más carga en la red implica una mayor carga computacional para el simulador de la red y un enlentecimiento en el tiempo de simulación, como se comprueba en la Figura 2.b. Ahora, con el router más detallado, el rendimiento cae un 40% en promedio. La implementación simple atenúa esta caída entre un 5 y un 10%. Bajo estas condiciones, en este protocolo exploramos optimizaciones de rendimiento adicionales. La interfaz adaptativa (denominada AI en los resultados) es capaz de mejorar significativamente el rendimiento de la simulación, reduciendo la diferencia con el simulador original de Ruby en más de un 10% y con un error por debajo del 2% para los dos tipos de encaminadores. Los datos suministrados se han obtenido empleando un umbral de 25 paquetes en la red antes de activar el simulador TOPAZ. Incluso con una pequeña red y un router simple, el desbalanceo entre la simulación de la red y el resto de los componentes del sistema hace interesante la simulación multithread (denominada P en los resultados), especialmente si se requiere ejecutar rápidamente una aplicación en particular o máxima precisión para el modelo de red.

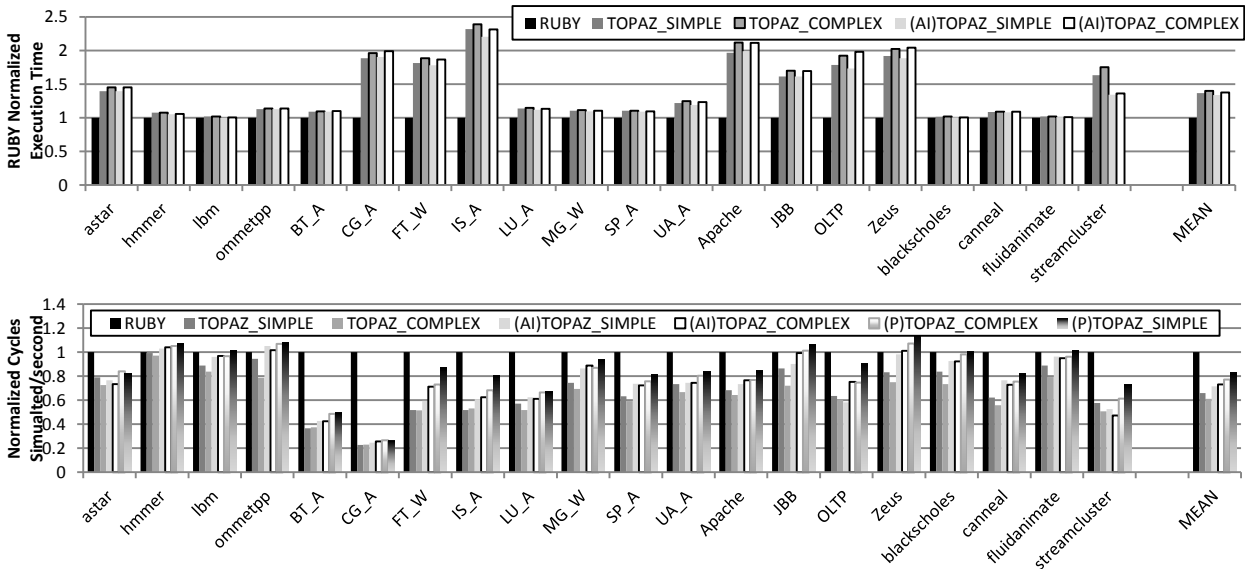

Figura 2. Protocolo de coherencia Broadcast. (a) Tiempo de ejecución. (b) Rendimiento de las diferentes herramientas de simulación.

IV. Evaluación de Redes de Interconexión de Supercomputadores

Esta clase de sistemas tiene la característica singular de un número elevado de encaminadores. El mayor desafío para una herramienta de simulación de este tipo de sistema aparece cuando es imprescindible un modelado preciso del encaminador para descubrir potenciales inestabilidades del sistema [32]. Los requerimientos de memoria para simular miles de nodos pueden ser altos. TOPAZ puede usar una asignación de memoria avanzada que acelera la ejecución y limita potenciales problemas durante la simulación. Adicionalmente, la implementación multithread permite sacar ventaja del actual predominio de los servidores multicore.

Para mostrar la escalabilidad de la herramienta, vamos a emplear un toro 3D con el modelo simple del Bubble Router [14], como en el caso de los sistemas Blue Gene de IBM [2]. Evaluamos el rendimiento obtenido con hasta un millón de routers en la red a fin de determinar la escalabilidad de la paralelización. La Figura 3 muestra los resultados obtenidos con un servidor con 12 cores y 54 Gbytes de memoria principal basado en el Xeon E5645. Con 32Krouters la simulación emplea aproximadamente 1.5GB de memoria, hasta los 49GB empleados para 1 millón de nodos, manteniendo una escalabilidad adecuada en todos los casos. A la vista de los resultados de speedup, TOPAZ también demuestra ser una herramienta apropiada para tratar con grandes sistemas.

Ejecutar la simulación para tales sistemas empleando simulaciones secuenciales sería prohibitivo debido a la masiva cantidad de memoria necesaria para abastecer a 12 o más cores. De acuerdo al tamaño del sistema, la escalabilidad y la memoria disponible en el servidor, el usuario podría seleccionar el número óptimo de threads por simulación. Por ejemplo, en nuestro caso, si el número de routers es 256K, el número óptimo de simulaciones a ejecutar en nuestro servidor es tres, dedicando a cada una 4 cores y tendríamos 36 GB de utilización de memoria y un speedup casi perfecto.

V. Conclusiones

Hemos presentado TOPAZ, que es una potente herramienta concebida para facilitar la investigación en redes de interconexión. Su integración con una de las plataformas más comunes en la evaluación de CMPs y su flexibilidad para simular redes de interconexión de gran escala podría convertirla en una herramienta atractiva para un amplio rango de usuarios. Continuaremos suministrando soporte de la herramienta ya que se trata de uno de nuestros principales recursos

de investigación. El carácter de código abierto también simplificará el que terceras partes realicen aportaciones.

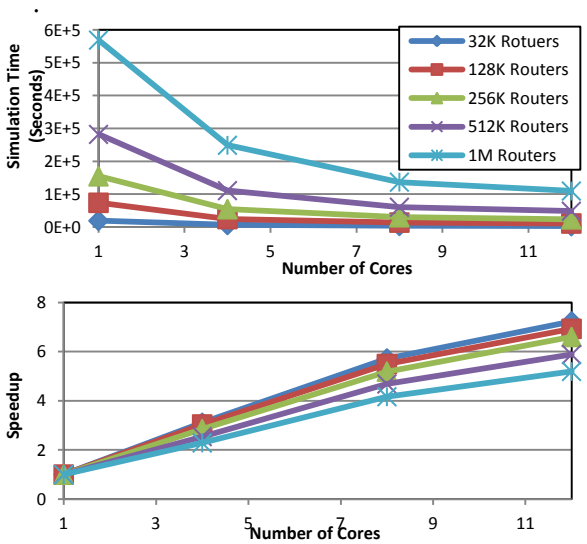

Figura 3. (a) Tiempo de simulación para 50K ciclos. (b) Speedup logrado.

VI. Agradecimientos

Los autores agradecen a José Ángel Herrero por su valiosa ayuda con el entorno de computación. Este trabajo ha sido financiado mediante el proyecto TIN2010-18159 del Ministerio de Ciencia e Innovación y por la red de excelencia europea HiPEAC.

VII. Referencias

[1] W. J. Dally and B. Towles, "Route packets, not wires: on-chip interconnection networks," *Proceedings of the 38th Design Automation Conference (IEEE Cat. No.01CH37232)*, pp. 684-689.

[2] N. R. Adiga et al., "Blue Gene/L torus interconnection network," *IBM Journal of Research and Development*, vol. 49, no. 2.3, pp. 265–276, Mar. 2005.

[3] R. Kalla, B. Sinharoy, W. J. Starke, and M. Floyd, "Power7: IBM's Next-Generation Server Processor," *IEEE Micro*, vol. 30, no. 2, pp. 7-15, 2010.

[4] C. Park et al., "A 1.2 TB/s on-chip ring interconnect for 45nm 8-core enterprise Xeon® processor," in *2010 IEEE International Solid-State Circuits Conference - (ISSCC)*, 2010, pp. 180-181.

[5] M. Coppola, R. Locatelli, G. Maruccia, L. Pieralisi, and A. Scandurra, "Spidergon: a novel on-chip communication network," in *International Symposium on System-on-Chip*, 2004, vol. 50, no. 2, pp. 15-22.

[6] N. Agarwal, T. Krishna, L.-S. Peh, and N. K. Jha, "GARNET: A detailed on-chip network model inside a full-system simulator," *2009 IEEE International Symposium on Performance Analysis of Systems and Software*, pp. 33-42, Apr. 2009.

[7] M. M. K. Martin et al., "Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset," *ACM SIGARCH Computer Architecture News*, vol. 33, no. 4, p. 99, 2005.

[8] J. Navaridas, J. Miguel-Alonso, J. a. Pascual, and F. J. Ridruejo, "Simulating and evaluating interconnection networks with INSEE," *Simulation Modelling Practice and Theory*, vol. 19, no. 1, pp. 494-515, Jan. 2011.

[9] F. Fazzino and M. Palesi, "Noxim: Network-on-chip simulator." [Online]. Available: http://sourceforge.net/projects/noxim.

[10] M. Rosenblum, S. A. Herrod, E. Witchel, and A. Gupta, "Complete computer system simulation: The SimOS approach," *Parallel & Distributed Technology: Systems & Applications, IEEE*, vol. 3, no. 4, pp. 34–43, 1995.

[11] N. Binkert et al., "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, p. 1, Aug. 2011.

[12] "Topaz Project page." [Online]. Available: http://code.google.com/p/tpzsimul/.

[13] V. Puente, J. A. Gregorio, and R. Beivide, *SICOSYS: an integrated framework for studying interconnection network performance in multiprocessor systems*. IEEE Comput. Soc, 2002, pp. 15-22.

[14] V. Puente, C. Izu, R. Beivide, J. A. Gregorio, F. Vallejo, and J. M. Prellezo, "The Adaptive Bubble Router," *Journal of Parallel and Distributed Computing*, vol. 61, no. 9, pp. 1180-1208, 2001.

[15] C. Carrion, R. Beivide, J. A. Gregorio, and F. Vallejo, "A flow control mechanism to avoid message deadlock in k-ary n-cube networks," in *Proceedings Fourth International Conference on High-Performance Computing*, 2001, pp. 322-329.

[16] W. J. Dally, "Virtual-channel flow control," *IEEE Transactions on Parallel and Distributed Systems*, vol. 3, no. 2, pp. 194-205, Mar. 1992.

[17] L.-S. Peh and W. J. Dally, "A delay model and speculative architecture for pipelined routers," in *Proceedings HPCA Seventh International Symposium on High-Performance Computer Architecture*, 2001, pp. 255-266.

[18] N. E. Jerger, L. S. Peh, and M. Lipasti, "Virtual circuit tree multicasting: A case for on-chip hardware multicast support," in *Computer Architecture, 2008. ISCA'08. 35th International Symposium on*, 2008, pp. 229-240.

[19] P. Abad, J. A. Gregorio, V. Puente, and P. Prieto, "Rotary router: an efficient architecture for CMP interconnection networks," in *International Symposium on Computer Architecture*, 2007, vol. 35, no. 2.

[20] P. Abad, V. Puente, and J. A. Gregorio, "MRR: Enabling fully adaptive multicast routing for CMP interconnection networks," in *2009 IEEE 15th International Symposium on High Performance Computer Architecture*, 2009, pp. 355-366.

[21] T. Moscibroda and O. Mutlu, "A case for bufferless routing in on-chip networks," *ACM SIGARCH Computer Architecture News*, vol. 37, no. 3, p. 196, Jun. 2009.

[22] Y.-C. Lan, H.-A. Lin, S.-H. Lo, Y. H. Hu, and S.-J. Chen, "A Bidirectional NoC (BiNoC) architecture with dynamic self-reconfigurable channel," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 30, no. 3, pp. 427-440, Mar. 2011.

[23] M. Katevenis, "Fast switching and fair control of congested flow in broadband networks," *IEEE Journal on Selected Areas in Communications*, vol. 5, no. 8, pp. 1315-1326, Oct. 1987.

[24] A. Kumary, P. Kunduz, A. P. Singhx, L.-S. Pehy, and N. K. Jhay, "A 4.6Tbits/s 3.6GHz single-cycle NoC router with a novel switch allocator in 65nm CMOS," *2007 25th International Conference on Computer Design*, pp. 63-70, Oct. 2007.

[25] N. Neelakantam, C. Blundell, J. Devietti, M. M. K. Martin, and C. Zilles, "FeS2: A Full-system execution-driven simulator for x86 Simulating an Application," in *Poster ASPLOS*, 2007.

[26] H. Jin, M. Frumkin, and J. Yan, "The OpenMP implementation of NAS parallel benchmarks and its performance," *NASA Ames Research Center," Technical Report NAS-99-011*, Citeseer, 1999.

[27] C. Bienia, S. Kumar, J. P. Singh, and K. Li, "The PARSEC benchmark suite," in *Proceedings of the 17th international conference on Parallel architectures and compilation techniques - PACT '08*, 2008, pp. 72-80.

[28] A. R. Alameldeen et al., "Simulating a $2 M Commercial Server on a $2 K PC," *Computer*, vol. 36, no. 2, pp. 50-57, 2003.

[29] V. Standard Performance Evaluation Corporation, SPEC*, http://www.spec.org, Warrenton, "SPEC 2006." .

[30] N. Kurd, J. Douglas, P. Mosalikanti, and R. Kumar, "Next generation Intel® micro-architecture (Nehalem) clocking architecture," in *2008 IEEE Symposium on VLSI Circuits*, 2008, pp. 62-63.

[31] M. Martin and M. Hill, "Token Coherence: a New Framework for Shared-Memory Multiprocessors," *Micro, IEEE*, pp. 108-116, 2004.

[32] J. Miguel-Alonso, J. Gregorio, V. Puente, F. Vallejo, and R. Beivide, "Load Unbalance in k-ary n-cube Networks," in *Euro-Par 2004 Parallel Processing*, 2004, pp. 900–907.