Acelerando las simulaciones de sistema completo usando Simics en sistemas multiprocesador

S. González[1], F. Triviño[2], F. J. Andujar[2], J. L. Sánchez[2], F. J. Alfaro[2]

Resumen— **El uso de simuladores de sistema completo para evaluar el desarrollo de un determinado componente en un sistema de computación, es una práctica frecuente. Esto permite obtener mejores resultados y conclusiones en comparación a una simulación parcial donde solo se modela una parte del sistema y se descarta su interacción con el resto del sistema.**

Sin embargo, realizar una simulación de sistema completo trae consigo inconvenientes como un mayor tiempo para realizar las pruebas y un mayor consumo de recursos. Esto puede generar una demanda creciente proporcional a la complejidad del modelo a analizar, como se da en la investigación de sistemas multiprocesador al aumentar el número de procesadores o al realizar simulaciones en paralelo.

En este trabajo se describen las mejoras de rendimiento en términos de tiempo de simulación que ofrece la versión 4 del simulador Simics, con la incorporación de nuevas tecnologías pensadas para mejorar el uso de sistemas multiprocesador, como es Simics Accelerator. Se realizan pruebas con el benchmark PARSEC, midiendo el tiempo de simulación y comparando el rendimiento con la versión 3 de Simics, analizando las ventajas y desventajas en optar por la actualización de versión.

Palabras clave— **Simulación, rendimiento, Simics4.**

I. Introducción

La simulación es un método muy usado en la investigación y diseño de propuestas de arquitectura de computadores al permitir evaluar una variedad de arquitecturas sin tener que construirlas, permitiendo la reducción en coste y tiempo de desarrollo de un proyecto. Además, con ello se mejoran los procedimientos de validación de nuevos modelos gracias a la sistematización de pruebas, las cuales pueden ser replicadas fácilmente y ejecutadas en paralelo, así como la obtención de resultados.

Muchas veces será útil realizar un análisis del comportamiento del sistema en su totalidad incluyendo el procesador, la memoria, los dispositivos de entrada y salida, buses, redes y otras interconexiones. Todo ello se puede evaluar de forma conjunta permitiendo la ejecución de un sistema operativo y otras aplicaciones comerciales como benchmarks. Esa es la finalidad con la que surgen los simuladores de sistema completo [1]. Un inconveniente de este tipo de simulaciones es que requieren de mucho tiempo para completarse, debido al gran nivel de detalle del sistema modelado. Así pues, un reto importante es poder reducir al máximo el tiempo de simulación.

[1]Dpto. de Informática, Univ. Peruana Cayetano Heredia, e-mail: santos.gonzalez.t@upch.pe

[2]Dpto. de Sistemas Informáticos,Univ. Castilla-La Mancha, e-mail: {ftrivino, fandujar, jsanchez, falfaro}@dsi.uclm.es

Simics [4] es una herramienta de simulación de sistema completo, capaz de modelar diferentes tipos de arquitecturas. Sin embargo, cuando el sistema es muy complejo las simulaciones pueden requerir horas o incluso días en completarse.

En este trabajo se revisa la versión 4 de Simics analizando sus nuevas características. Éstas surgen como resultado de las nuevas tendencias en el uso de máquinas multiprocesador y clusters. Una de las más importantes es la inclusión de Simics Accelerator [2], que permite reducir el tiempo de ejecución haciendo un uso más eficiente del hardware donde se realiza la simulación.

Para determinar cómo influyen las nuevas características de Simics 4, se realizarán diversas simulaciones usando el benchmark PARSEC [3], el cual agrupa un largo y variado conjunto de aplicaciones que han sido correctamente paralelizadas con diferentes técnicas. Se trata de ofrecer argumentos para poder decidir sobre la actualización o no del sistema y dependiendo del modelo a simular poder tomar una decisión.

Este artículo está organizado de la siguiente manera: en la Sección II se incluye una breve descripción del simulador Simics 4.2 y de sus características más importantes. En la Sección III se describe el proceso de pruebas para obtener los tiempos de simulación. Finalmente, en la Sección IV se presentan las conclusiones y trabajo futuro.

II. SIMICS 4

Simics [4] es una plataforma que permite simular un sistema completo lo cual facilita tanto el desarrollo de hardware como de software proporcionando lo necesario para la simulación de ambos componentes dentro de un mismo contexto. Como se puede observar en la Figura 1 tanto los requerimientos hardware como los del sistema, cuyo modelado mantiene una estructura modular, son completamente simulados en una máquina (host) [1].

También se pueden tener varias arquitecturas tanto monoprocesador como multiprocesador y ejecutar en ellas sistemas operativos convencionales, benchmarks, aplicaciones de escritorio, entre otras aplicaciones comerciales. Una de las ventajas que se tiene es que se pueden usar cargas de trabajo reales, algo que no se puede hacer con otros simuladores.

La versión utilizada para el presente trabajo es la versión 4.2. Entre las principales características con las que cuenta esta versión se pueden destacar:

- Mejoras en rendimiento y escalabilidad debido a la inclusión de Simics Accelerator 2.0. Permite ejecutar simulaciones de sistemas distribuidos y máquinas multiprocesador acelerando el desarrollo de software y hardware ayudados también con las herramientas de depuración y checkpointing.
- Mejoras en los modelos a simular, permitiendo la comprensión de los programas a través de un visualizador de rendimiento y facilitando el diagnóstico de errores.
- Mejoras en la interfaz de usuario logrando integrarse con Eclipse a través de plugins para iniciar y controlar las sesiones de Simics. Esto hace útil las herramientas y flujos de trabajo de Eclipse.
- Conectividad a través de telnet, un visor de memoria y soporte unicode.

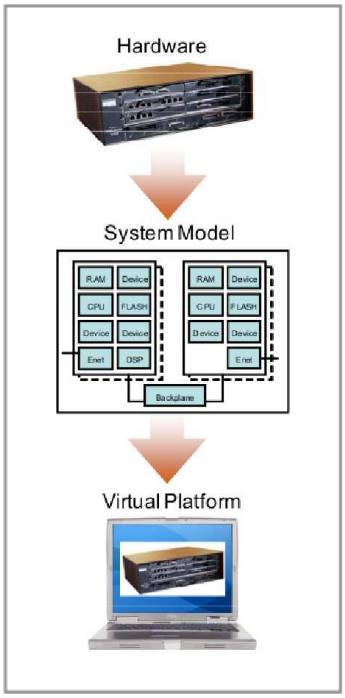

Fig. 1. Simulador de sistema completo.

Simics 4 soporta nuevos tipos de arquitecturas. En la Tabla I se pueden observar los nuevos modelos incorporados para esta versión [4]. Simics, dentro de sus nuevas características, también permite realizar simulaciones híbridas logrando unir modelos de procesadores de distintos niveles dentro de un sistema simulado sirviendo tanto para el desarrollo rápido como para su validación, y luego poder obtener un análisis detallado de rendimiento. Este tipo de simulaciones híbridas están disponibles en los modelos de microprocesador Freescale QorIQ P4080, y en próximas versiones se podrá tener un API genérico para que pueda ser aplicado a otras microarquitecturas.

De igual forma, pueden usarse nuevas extensiones para Simics como es el soporte en tiempo real del sistema operativo. Así, se permite que Simics pueda detectar cuándo un proceso se inicia, termina y se mantiene activo en el sistema simulado, sirviendo como herramienta para la detección de errores.

TABLA I
NUEVAS ARQUITECTURAS SOPORTADAS POR SIMICS 4.

Modelos	**Procesadores**	**Componentes**
IBM PowerPC 464	PPC464FP	AMCC PPC440GX SoC, Memoria DDR, FLASH, Conectividad serial y Ethernet
Freescale Power QUICC II Pro MPC83xx	MPC8347, MPC8360E	Memoria DDR, FLASH, Conectividad serial y Ethernet
ARM Integrator/CP	ARM926, ARM1136, ARM1176	Memoria DDR, FLASH, Conectividad serial y Ethernet
ARM Basic	Intel Strong ARM	RAM, Conectividad serial

Otra de las características que incorpora Simics es la de lograr un puente con los modelos de SystemC permitiendo que éstos sean incluidos como parte del sistema de simulación de Simics. De esta forma, será posible construir una plataforma virtual que permita ser ejecutada a través de la interacción de modelos en DML, C, Python y SystemC logrando que los usuarios puedan construir de manera rápida una plataforma completa, al facilitar la reutilización de estructuras previamente realizadas sin necesidad de tener que hacer cambios.

También es posible trabajar con TLM (Transaction-level modeling) [5] que tiene la propiedad de realizar la comunicación a través de llamadas a funciones directamente entre los módulos simulados sin tener que interactuar con el kernel simulado. Esto se realiza en unidades que sean lo más largas posibles, para reducir el número de comunicaciones, y con la menor frecuencia posible.

Las versiones anteriores de Simics basaban el rendimiento y escalabilidad con el módulo llamado Simics Central [6]. Este módulo administraba la conexión de nodos heterogéneos de distintas máquinas simuladas, las cuales podían encontrarse en distintos hosts. Así pues, dicho módulo tenía que encargarse de sincronizar el tiempo virtual entre el simulador y el tráfico simulado entre las máquinas. Ahora la nueva versión de Simics reemplaza este módulo de simulación distribuida incorporando Simics Accelerator 2.0. Este nuevo módulo trae consigo otras venta-

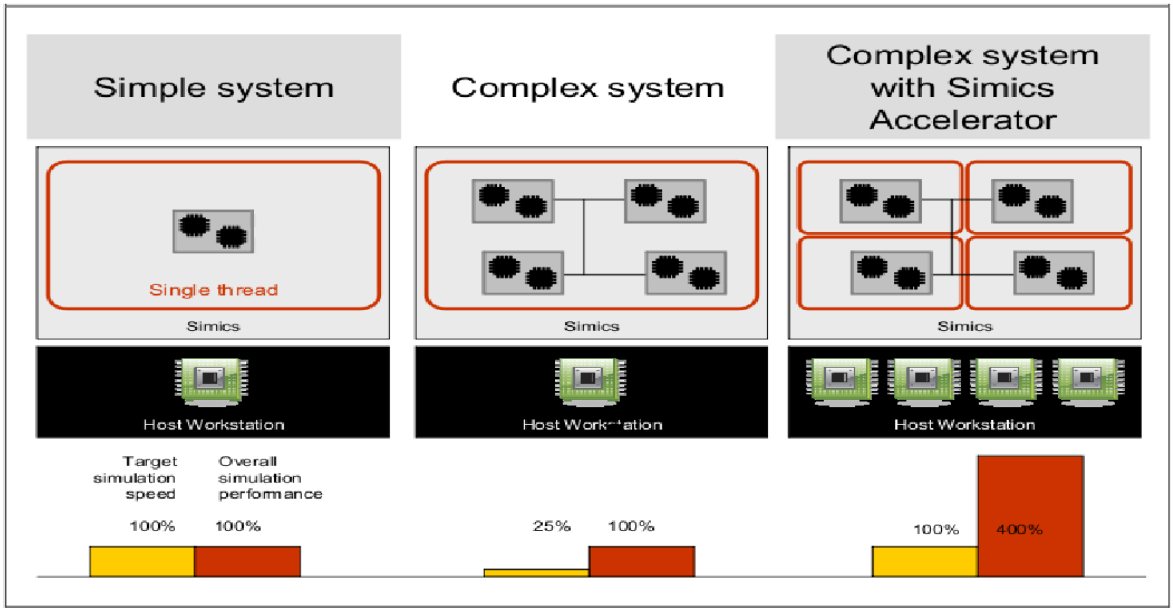

Fig. 2. Simulación Multihilo[2]

jas adicionales que repercuten en el rendimiento del sistema. A continuación se resumen las principales ventajas que mejoran la velocidad en la simulación.

A. Simics Accelerator 2.0

Uno de los principales requisitos al realizar simulaciones es que éstas, además de precisas, puedan ser realizadas en el menor tiempo posible. Esto debería cumplirse incluso simulando sistemas y cargas de trabajo grandes sacando provecho del uso de servidores y clusters.

Simics busca este objetivo a través de este nuevo módulo donde se implementan mejoras relacionadas con la decisión en el nivel correcto de abstracción del tiempo, en la metodología de modelado basada en transacciones entre los distintos dispositivos, creación de modelos de procesadores rápidos y simulaciones multihilo [1]. Estas últimas están orientadas a ser aplicadas en la ejecución de modelos más complejos, como se muestra en la Figura 2. Empezando con un modelo sencillo se simula una máquina en una computadora de un solo núcleo. El host provee cierto porcentaje de procesamiento, lo que equivale a la velocidad de simulación en términos del rendimiento del sistema global y de cómo es percibido por el usuario.

Cuando se realiza un modelo más complejo, de 4 máquinas simuladas en el mismo host con el mismo poder de procesamiento, el rendimiento esta vez tiene que ser dividido por la cantidad de máquinas simuladas. Esta vez el usuario percibirá una reducción de cuatro veces la velocidad de simulación. Usando Simics Accelerator y un host con cuatro núcleos, el mismo modelo complejo de cuatro máquinas puede aprovechar la capacidad de procesamiento, ahora mayor en el host, logrando que cada procesador del host pueda realizar el procesamiento de cada máquina simulada. Esto producirá para el usuario una percepción de la misma velocidad que en el caso inicial, como el de una sola máquina simulada en un computador de un solo núcleo.

La memoria compartida es uno de los medios que usa Simics para liberar la demanda de memoria en el sistema a simular. Lo que hace es verificar que los contenidos dentro de las memorias RAM, ROM, FLASH o del disco simulado no sean redundantes de tal forma que no se duplique contenido y sólo se tenga una copia del mismo en la memoria de la máquina en donde se realiza la simulación.

La simulación distribuida que ofrece esta nueva versión de Simics permite el uso de múltiples hosts para aumentar la escalabilidad y aprovechar mejor el uso de clusters. Esto mejora directamente el rendimiento en la ejecución de múltiples sesiones de Simics en paralelo, especificándose la cantidad de núcleos que serán usados para cada sesión de Simics, logrando de esta forma escalabilidad, particionando los recursos del host para la simulación y evitando bloqueos entre las simulaciones.

La sincronización es importante cuando se realiza una simulación distribuida. Idealmente los procesos podrían realizarse simultáneamente en tiempo simulado y en tiempo real pero eso no se puede por la gran cantidad de sobrecarga que sería necesaria para la sincronización dejando muy poco tiempo para el trabajo real. Simics introduce un pequeño retardo que hace que la simulación distribuida no esté completamente sincronizada, a costa de no producir tanta sobrecarga permitiendo que los distintos componentes puedan comunicarse solamente a intervalos especificados, los cuales son definidos desde la configuración de la simulación.

Simics permite la simulación de máquinas interconectadas mediante una red de área local lográndose conectar varias máquinas mediante ethernet-links, con las que se modela una red Ethernet a nivel de trama. Esta conexión se puede ver como un cable ethernet que va conectado a un dispositivo ethernet

de la máquina virtual o como un switch/hub al que pueden conectarse varios dispositivos. El tráfico que se envía sobre dicha conexión puede ser TCP/IP o cualquier otro protocolo que funcione en Ethernet.

A esta conexión también se le puede añadir ciertos servicios IP simulados mediante una clase llamada service-node que proporciona un nodo de red simulado a modo de servidor y que puede actuar como un router IP entre redes.

Simics permite mejoras en el modelado de sistemas, lo cual es de suma importancia para obtener mayores beneficios de la simulación, como se puede ver en la Figura 3, donde se representan un cluster de procesadores, SoCS, memorias, placa y demás partes de un sistema interconectados para luego ser combinados a través de una estructura jerárquica. Todo ello usando las herramientas de diseño de dispositivos con los que cuenta Simics Accelerator.

Para poder realizar simulaciones en paralelo se debe cumplir con ciertos requisitos como que los modelos simulados no deben compartir memoria y que las arquitecturas a simular permitan simulación multihilo. Dentro de los modelos ya diseñados, los que permiten simulación multihilo son el x86-440bx y MPC8641.

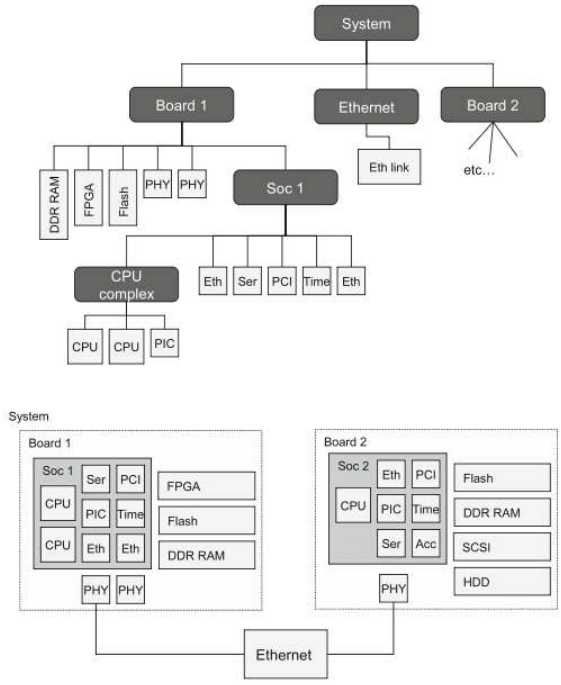

Fig. 3. Ejemplo de componentes jerárquicos.

B. Simics 4 y GEMS

Una de las razones para comparar las versiones de Simics, es para poder determinar si es conveniente o no realizar la actualización a la versión 4 en los trabajos ya desarrollados con la versión 3, teniendo en cuenta, por ejemplo, la compatibilidad con otras herramientas que extienden Simics como es el caso de GEMS (General Execution-driven Multiprocessor Simulator) [7]. GEMS es un programa compuesto por un conjunto de módulos que hace posible la simulación de sistemas multiprocesadores de forma más detallada.

Básicamente, GEMS está formado por dos módulos: Ruby y Opal. Ruby permite simular la jerarquía de memorias cache, controladores de memoria, bancos de memoria principal y la red que interconecta todos estos elementos. Opal permite la simulación con ejecución fuera de orden, para modelar diferentes arquitecturas, desde monoprocesadores hasta multiprocesadores como SMPs, CC-NUMAs y CMPs, donde es posible la ejecución de varios hilos de forma simultánea.

Si bien originalmente fue desarrollado para el estudio de sistemas de memoria proporcionando modelos de temporización detallados, enfocados a la simulación de arquitecturas concretas, ahora también permite profundizar en otros tipo de investigaciones como las de redes de interconexión en el chip [8].

El inconveniente que por ahora mantiene Simics4 es no ser compatible con la actual versión de GEMS 2.1.1. En la página de GEMS se cuenta con un parche externo que funciona con las versiones 3.0, 3.2, 4.0, 4.2, y 4.4. Sin embargo, también se hace mención que éste no ofrece soporte para el módulo Opal y se retira el soporte para la versión Simics2.2, y en caso de que se desee trabajar con estos módulos se sugiere usar la versión de Simics3 [9]. También se indica que el parche no ha sido probado por el equipo de GEMS y no se brindará soporte para el mismo. Según la información que se tiene actualmente en los foros y la documentación hay bastantes problemas para la integración de GEMS con Simics4. Recientemente, GEMS informó de su integración con el simulador M5 [10], otro simulador de sistema completo, el cual ahora se encuentra integrado en un nuevo simulador llamado GEM5, por lo cual aparentemente GEMS estaría más abocado a dicha integración, dejando un poco de lado su continuidad con Simics.

III. Evaluación de Rendimiento

En este trabajo se busca comparar el tiempo de simulación entre las versiones 3 y 4 de Simics, y cómo influye la complejidad del modelado al aumentar la cantidad de procesadores en dicho cálculo.

A. Arquitectura modelada

Para la realización de las pruebas se usó la arquitectura x86-440BX, la cual puede modelar varios sistemas con procesadores x86 y AMD64 basados en el chipset 440BX. “Tango” es la máquina simulada, que viene ya instalada con Fedora Core 5 con soporte para desarrolladores.

En la Tabla II se pueden ver las principales características de la máquina simulada utilizada para las pruebas. Por el tipo de arquitectura se tiene el límite de un máximo de ocho procesadores en la placa base. En las pruebas se harán mediciones de tiempo de simulación en donde se variará la cantidad de procesadores.

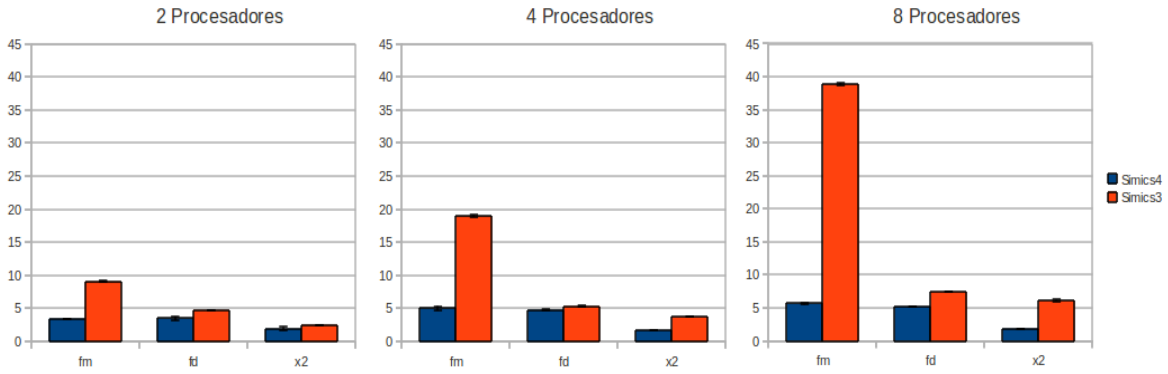

Fig. 4. Resultados de tiempo de simulación de aplicaciones PARSEC

TABLA II
CARACTERÍSTICAS DE LA MÁQUINA SIMULADA.

Modelo	x86-440BX
Slots para CPU	8 máximo.
Dump de disco	tango1-fedora5.craff
Kernel	2.6.15
Procesador	2 GHz Pentium 4
Disco duro	20GB
Memoria RAM	2 GB

B. *Aplicaciones PARSEC*

Para la evaluación de rendimiento se han realizado pruebas con el benchmark PARSEC, midiendo el tiempo de simulación al ejecutar las aplicaciones que forman esta suite. Para poder realizar la medición se hace uso de las Instrucciones Mágicas (Magic Instructions), para lo cual se compiló el benchmark PARSEC con la opción gcc-hooks y determinar así el momento en que entra a una fase la ejecución de la aplicación.

Dependiendo de la arquitectura, se podrá pasar un valor a través de las instrucciones mágicas, teniendo como referencia el archivo magic-instruction.h donde se detalla las instrucciones para cada tipo de arquitectura y para el caso de la arquitectura X86 en Simics3 solo se retorna el valor 0 por lo que se tendrá que pasar el valor a través de los registros del procesador de la máquina simulada [11].

De todos los programas de la suite PARSEC, aquí se mostrarán resultados de las aplicaciones freqmine (fm), fluidanimate (fa) y x264 (x2).

C. *Realización de pruebas*

Para las pruebas se crearon checkpoints base con una cantidad determinada de procesadores para cada prueba. Se simularon máquinas con 2, 4 y 8 procesadores. Teniendo en cuenta las diferencias en cómo se pasan las instrucciones mágicas en las dos versiones de Simics, para cada una de ellas se compiló el benchmark con las instrucciones mágicas correspondientes.

Una vez compilado el benchmark a través del API de Simics y con un script en python se recogen las estadísticas del tiempo de simulación de los distintos programas usados. También hay que tener en cuenta, de igual forma que para la instrucciones mágicas, que algunas funciones ya no son soportadas por el nuevo API, como es el caso de la función run(), que se usaba desde la versión de Simics 2 y que aún era compatible para la versión de Simics 3. La versión 4 solo soporta la función SIM_run_command().

Para la creación de los checkpoints se ha de tener en cuenta que no son compatibles para las versiones anteriores, lo que quiere decir que en caso de que creemos un checkpoint con la versión de Simics 4 no podrá ser leída con la de Simics 3. Otro punto a tener en cuenta es que la nueva versión de Simics asigna un código al momento de crear el checkpoint que se guarda en la variable build_id en el archivo de configuración. Al migrar un checkpoint este código puede dar problemas al no identificar la misma versión por lo que en caso de que sea necesario tendrá que ser modificado.

Las imágenes del disco trabajadas entre distintas versiones pueden ser reutilizadas entre versiones, de igual forma las que han sido agrupadas con las diferencias de disco con la herramienta craff de Simics. Lo único que se tiene que tener en cuenta es la compatibilidad con las instrucciones mágicas de cada versión.

D. *Resultados*

Una vez realizadas las ejecuciones de las aplicaciones se obtuvieron los tiempos de simulación por cada una. En la Figura 4 se puede ver el tiempo de simulación para las dos versiones de Simics a estudiar, y para las aplicaciones utilizadas. En las gráficas se puede apreciar que las simulaciones realizadas con la nueva versión de Simics se completan con un menor tiempo de simulación que con la versión anterior. En general, y como parece lógico, al aumentar la cantidad de procesadores el tiempo de simulación también aumenta, lo que se puede apreciar más claramente para la versión de Simics 3.

De las pruebas realizadas, se obtiene un mayor rango de diferencia en el tiempo de simulación para el caso de la aplicación freqmine. Para este caso se obtiene una reducción del tiempo de simulación de 61 % para 2 procesadores, 70,1 % para 4 procesadores y

85,4 % para 8 procesadores.

Por contra, con la aplicación x264 es en donde la diferencia es menor. En este caso se obtiene un 22,6 % para 2 procesadores, 53,9 % para 4 procesadores y 69,6 % para 8 procesadores.

En la Figura 5 se puede observar el tiempo total de simulación para las pruebas realizadas, es decir el tiempo que tomaría haber ejecutado las 3 aplicaciones de la suite simultáneamente. En este caso tenemos una reducción del tiempo de simulación de 45 % cuando son 2 procesadores, 58 % para 4 procesadores y 75 % para 8 procesadores.

Para las pruebas con 8 procesadores se obtiene cerca de un 10 % de mejor rendimiento por cada procesador. En promedio, se puede alcanzar un 60 % de reducción en tiempo de simulación respecto a la versión 3.

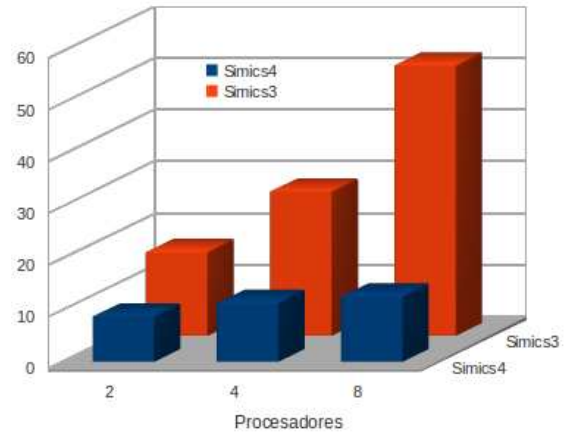

Fig. 5. Tiempos totales de simulación.

IV. Conclusiones

Las mejoras en cuanto a aceleración de procesos y tiempo de simulación para la nueva versión de Simics eliminan cuellos de botella y mejoran el rendimiento en modelos complejos, realizando varias simulaciones en paralelo y aprovechando el poder de procesamiento de un host multinúcleo o incluso un cluster.

Las mejoras en cuanto a sincronización de tiempos en la simulación y la administración de información redundante al momento de ejecutar el benchmark llegan a reducir en promedio un 60 % de tiempo de simulación para las pruebas realizadas con las aplicaciones freqmine, fluidanimate y x264 del PARSEC. Y para el caso de una simulación con una máquina con 8 procesadores se llega a obtener hasta un 10 % de reducción en tiempo de simulación por cada procesador.

En general, las mejoras obtenidas por la nueva versión de Simics4 son muy positivas permitiendo realizar simulaciones con una mayor cantidad de procesadores sin tener una gran demanda en tiempo de simulación, permitiendo así analizar modelos más complejos.

El punto débil por el momento de la nueva versión de Simics es la no compatibilidad con otros programas que lo extienden, como es el caso de GEMS, el cual es compatible con la versión 3 de Simics y actualmente no ofrece soporte para la nueva versión. Además, es posible que no lo tenga en cuenta por su actual trabajo en el nuevo simulador GEM5

En el momento de redactar este documento, se estaba indicando un estudio para mejorar las simulaciones con paralelismo de núcleos y su análisis, los cuales dependían del uso de GEMS pero al tener problemas para la compilación de sus módulos se podría tener en cuenta para un futuro trabajo el análisis del nuevo simulador para las investigaciones con máquinas multinúcleo.

Agradecimientos

Este trabajo ha sido realizado gracias a la beca de ayudas Universidad de Castilla-La Mancha - Banco Santander para estancias de investigación de profesores iberoamericanos en la Universidad de Castilla-La Mancha en 2011

Referencias

[1] Jakob Engblom, Daniel Aarno, and Bengt Werner *Full-System Simulation from Embedded to High-Performance Systemsl*, Processor and System-on-Chip Simulation, pp. 25-44 , 2010.

[2] Jakob Engblom, *Simics Accelerator*, in Whitepaper Virtutech, March 2009.

[3] Christian Bienia, Sanjeev Kumar, Jaswinder Pal Singh,and Kai Li, *The PARSEC Benchmark Suite: Characterization and architectural implications*, in Proceedings of the 17th International Conference on Parallel Architectures and Compilation Techniques, October 2008

[4] Simics Models, http://www.virtutech.com/products /simicsmodels

[5] Jakob Engblom, *Transaction-Level Modeling in Simic*, in Whitepaper Virtutech, August 2009.

[6] Magnusson, P., Christensson, M., Eskilson, J., Forsgren, D., Hallberg, G., Hogberg, J.,Larsson, F., Moestedt, A., Werner, B. *Simics: A full system simulation platform*, Computer, Innovative Technology for Computer Professionals, pp. 50-58, Febrary 2002.

[7] Milo M. K. Martin, Daniel J. Sorin, Bradford M. Beckmann, Michael R. Marty, Min Xu, Alaa R. Alameldeen, Kevin E. Moore, Mark D. Hill, and David A. Wood, *Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset,*, SIGARCH Comput. Archit.News, vol. 33, no. 4, pp. 92-99, 2005.

[8] Francisco Triviño, Francisco J. Andujar, Alberto Ros, José L. Sánchez, Francisco J. Alfaro *Sistema Integrado de Simulación de NoCs*, XX Jornadas de Paralelismo La Coruña(Spain). Septiembre 2009.

[9] Multifacet GEMS: External patches for Simics, http://www.cs.wisc.edu/gems/common/release_notes/gems2.1.1_patch1_releasenotes.txt

[10] The GEM5 Simulator System, http://gem5.org/

[11] Virtutech: Simics User Guide for Unix, pp 143-145, Febrary 2008. https://www.simics.net/pub/simics/3.0_fyr609/simics-user-guide-unix.pdf