Una Nueva Metodología para el Estudio de Procesadores Realistas en las Titulaciones de Informática

C. Gómez[1], M. Serrano[2], M.E. Gómez[2] y J. Sahuquillo[2]

Resumen— **La arquitectura de los procesadores actuales ha evolucionado de forma vertiginosa en los últimos años, desembocando en el desarrollo de procesadores con varios núcleos computacionales capaces de dar soporte a varios hilos. La complejidad resultante dificulta la docencia y en la mayoría de las universidades se estudia de manera muy superficial los procesadores superescalares o multihilo, sobre todo en lo que respecta a la parte práctica de las asignaturas. Este hecho hace que los alumnos tengan una concepción un tanto alejada del funcionamiento real de un procesador multinúcleo actual. Por ello, en este artículo se presenta la metodología puesta en marcha durante este curso en la Universitat Politècnica de València para la enseñanza práctica de la Arquitectura de Computadores con un enfoque realista y actualizado. La metodología se basa en el uso de un simulador muy completo que permite simular todos los aspectos más interesantes de los procesadores actuales. Debido a la complejidad del simulador, éste se utiliza en distintas asignaturas y de forma progresiva, lo que permite a los alumnos llegar a un aprendizaje sólido de los procesadores actuales, incluida la red que interconecta los distintos núcleos. Los resultados nos han demostrado que el alumno, en un tiempo razonable, es capaz de manejarse con el simulador y entender conceptos complejos de los procesadores. Además algunos de ellos han sido capaces de obtener resultados interesantes que se han publicado en congresos de investigación de relevancia internacional.**

Palabras clave— **Docencia, arquitectura de computadores, simuladores, procesadores multicore.**

I. Introducción

En la última década, los dispositivos electrónicos han evolucionado de manera vertiginosa y su uso se ha extendido a todos los aspectos de nuestra vida. Hoy en día la mayoría de los ciudadanos disponen de teléfonos móviles inteligentes, ordenadores portátiles, televisores programables, etc. Esto ha sido posible, entre otros, gracias a la gran potencia computacional y el bajo consumo requerido por los procesadores modernos.

Esta potencia computacional ha venido acompañada de un incremento en la complejidad tanto hardware como funcional de los procesadores. A principios de los 2000 los procesadores monolíticos alcanzaban su máxima expresión con procesadores superescalares y ejecución fuera de orden que incluían complejos predictores de saltos y cachés complejas como la trace caché, soporte para la ejecución de varios hilos de manera simultánea, etc. Posteriormente, y debido entre otros, a problemas de consumo de potencia y temperatura, la industria empezó a diseñar procesadores con varios núcleos computacionales o procesadores multicore. En los que a su vez cada uno de estos núcleos es capaz de dar soporte a varios hilos de ejecución independientes. Debido a los requerimientos computacionales de los usuarios, el número de núcleos va creciendo de forma continuada, lo que requiere una interconexión rápida y eficiente desde el punto de vista energético para interconectar los distintos núcleos entre sí y con la memoria principal.

Esta gran complejidad y continua evolución exige al profesorado de Ingeniería Informática un esfuerzo considerable para estar al día e incluso comprender los diseños comerciales. Por contra, la complejidad dificulta la docencia. En la mayoría de las universidades se suele empezar estudiando el pipeline de un procesador escalar sencillo en los primeros cursos. Posteriormente, se suelen introducir los conceptos de procesador superescalar, y algunas veces, de multihilo. Este hecho provoca que los alumnos tengan una concepción alejada del funcionamiento real de un procesador multinúcleo actual.

En este trabajo se presenta una nueva metodología con distintos grados de dificultad que facilita al alumno de Ingeniería Informática tener un amplio y profundo conocimiento de los procesadores comerciales. Para ello, nos basamos en el uso de un simulador detallado ciclo-a-ciclo de procesadores multinúcleo y multihilo de uso difundido entre estudiantes de doctorado e investigadores de todo el mundo así como por empresas de diseño de procesadores. La metodología propone el uso del entorno de simulación en distintas asignaturas y dada su complejidad se propone su uso de forma progresiva empezando con un uso muy guíado sin contacto con el código en las primeras etapas y aumentando el grado de implicación por parte del alumno en trabajos de asignatura o proyectos fin de carrera.

El resto de este trabajo se organiza como sigue. La sección II describe como se aborda el estudio de la arquitectura del computador en la Universitat Politècnica de València y presenta el simulador que se ha utilizado durante el presente curso académico para acercar su estudio a la realidad. La sección III describe la metodología propuesta utilizando el simulador para obtener los resultados de aprendizaje que se persiguen. En la sección IV se presenta un caso de estudio. Finalmente, presentamos algunas conclusiones en la sección V.

[1]DSI, Universidad de Castilla-La mancha, e-mail: `Crispin.Gomez@uclm.es`.
[2]DISCA, Universitat Politècnica de València

II. Docencia de arquitectura de computadores

A. Arquitectura de computadores en la UPV

En esta sección se describe el estudio de los procesadores actuales en el plan de estudios de la Universitat Politècnica de València. Su estudio se cubre de manera incremental en distintas asignaturas. En segundo curso en la asignatura Estructura de Computadores se estudia un procesador con un pipeline sencillo y en la asignatura Arquitectura e Ingeniería de Computadores de cuarto curso se amplía para dar soporte a la ejecución especulativa. Sin embargo, en ambas asignaturas el enfoque es pedagógico para aclarar y definir los conceptos más básicos. Este enfoque es la base para en asignaturas posteriores proporcionar una formación mucho más amplia, aunque sea a costa de simplificaciones que hagan que el procesador explicado difiera de los procesadores actuales.

Las asignaturas que tienen una aproximación más cercana a los procesadores comerciales recientes son las asignaturas: Arquitecturas Avanzadas de Procesadores de quinto curso de la Intensificación de Ingeniería de Computadores y ofertada también en el Máster de Ingeniería de Computadores y la asignatura Redes en Chip del Máster en Ingeniería de Computadores. Ambas asignaturas las pueden cursar aquellos alumnos interesados en tener una visión más exacta del funcionamiento de los procesadores actuales. Se trata de dos asignaturas complementarias ya que es imposible entender el impacto en las prestaciones de los procesadores comerciales actuales considerando solo una de ellas sin tener en cuenta la otra. Por este motivo, durante el último curso académico hemos puesto en marcha un enfoque metodológico integral que abarque ambas asignaturas. El enfoque tiene dos importantes beneficios: por una parte, el alumno ve las prestaciones del sistema de una manera global; por otra, el alumno solo necesita aprender una única herramienta que va aprendiendo de forma incremental, dada la complejidad del simulador.

B. Problemática: distanciamiento entre teoría y práctica

La continua evolución de los procesadores ha hecho que la docencia haya tenido que actualizarse para acercarse, en la medida de lo posible, a los productos comerciales de forma continuada, a fin de que los alumnos consigan una formación relativamente actualizada. Esta actualización de la docencia requiere de adaptaciones tanto en el contenido teórico impartido en el aula como en los contenidos prácticos ejercitados en el laboratorio.

De cara a los contenidos abordados en el laboratorio se suelen utilizar distintos simuladores según el curso y los objetivos perseguidos. Entre los simuladores típicos podemos encontrar el DLXV para el estudio del pipeline de un procesador: ruta de datos, ver como avanzan las instrucciones por el pipeline, resolución de riesgos, etc. Se trata de un simulador relativamente sencillo de aprender y utilizar lo que facilita la labor de aprendizaje, además de útil para facilitar la comprensión de procesadores escalares simples que son la base de los procesadores superescalares actuales. Sin embargo el DLXV no puede proporcionar apoyo docente para el estudio de procesadores monolíticos superescalares o multihilo debido a que no es capaz de modelarlos. Este hecho provoca un distanciamiento importante entre los contenidos teóricos y los prácticos, dejando contenidos teóricos sin ser desarrollados de forma práctica en el laboratorio, lo cual, como es bien sabido ayuda a afianzar los conceptos. La mayoría de las universidades no resuelven este vacío y se deja el uso de simuladores más complejos, como el SimpleScalar [1] o el Multi2Sim [3] que permiten simular estas características, exclusivamente a los estudiantes de doctorado. Este hecho es claramente un error, no porque el alumno no utiliza dichos simuladores, sino porque se pierde el apoyo de dichas herramientas para afianzar y conseguir un conocimiento sólido de conceptos complejos que desarrollados en el laboratorio pueden facilitar su compresión.

La metodología propuesta no solo resuelve el problema en sus primeras etapas sino que al hacerlo de manera gradual motiva el interés del alumno al estudio de la arquitectura de computadores. A continuación se describen las características del simulador utilizado para implementar la metodología.

C. Simulador usado

El simulador utilizado debe ser capaz de modelar los procesadores recientes así como permitir al alumno interesado realizar la tesis doctoral. Por ello, hemos elegido Multi2sim, entre cuyas características destacan la capacidad para modelar procesadores *pipeline* superescalares, arquitecturas multihilo y multinúcleo, unidades de procesamiento gráfico (GPUs), y soporte para los benchmarks más comunes en investigación. Se trata de un simulador de tipo *application-only* que permite ejecutar una o más aplicaciones sin tener que arrancar primero un sistema operativo. A todo lo anterior hay que añadir que se trata de un proyecto de código abierto, que se puede descargar de [5].

El paradigma de simulación usado en este simulador se puede dividir en dos módulos: la simulación funcional y la detallada. La funcional es solo una emulación del programa que recibe como entrada, y tiene el mismo comportamiento que tendría dicho programa ejecutado nativamente en una máquina x86. El simulador detallado ofrece un modelo de estructuras de CPU como cachés, unidades funcionales, redes de interconexión, etc., y un *pipeline* con seis etapas (*fetch*, *decode*, *dispatch*, *issue*, *writeback* y *commit*), ofreciendo soporte para ejecución especulativa.

El número de hilos y el número de núcleos

es configurable. Respecto al soporte de ejecución multihilo, se pueden obtener diseños multihilo de grano grueso (CGMT), de grano fino (FGMT), y multihilo simultáneo (SMT).

La configuración de la jerarquía de memoria es muy flexible. Se puede usar cualquier número de niveles de caché, con cualquier número de cachés en cada nivel (se mantiene la coherencia mediante el protocolo MOESI). Las cachés pueden ser unificadas o separadas para datos e instrucciones, privadas o compartidas por núcleo, por hilo, o por unidad de cómputo de GPU, y pueden servir rangos específicos de direcciones físicas.

El modelo de red de interconexión entre diferentes niveles de la jerarquía de memoria también es ampliamente configurable. Incluye un conjunto de nodos finales, de switches y de enlaces, y una tabla de encaminamiento bidimensional.

Por otra parte, se puede obtener un informe detallado de las estadísticas relacionadas con el *pipeline* del procesador, clasificadas por hilo y por núcleo de ejecución. Hay resultados de simulación genéricos, y estadísticas tanto para las estructuras hardware (buffer de reordenación, cola de instrucciones, archivo de registros, buffer de destino de salto), como para cada etapa del *pipeline*.

El informe de estadísticas de jerarquía de memoria contiene una sección por cada caché, módulo de memoria principal y red de interconexión. Para cada módulo de memoria principal o caché, el informe muestra información de accesos, fallos, aciertos, lecturas, escrituras, etc. Así mismo, el informe de la simulación de la red de interconexión está formado por variables como las transferencias, los mensajes, el ancho de banda, la utilización, etc.

El entorno de simulación proporciona herramientas de depuración amigables y de fácil uso. Por ejemplo, el depurador del *pipeline* es una herramienta del paquete del simulador que se usa para ver gráficamente los diagramas de tiempo para simulaciones detalladas, mediante el cual se puede ver la traza ciclo a ciclo de una determinada etapa con una representación de las uops y macroinstrucciones ejecutadas.

Además, desde la versión 3.0, Multi2Sim ha aumentado sus capacidades de simulación con un modelo para las GPUs de AMD, que también cuenta con su propio depurador de *pipeline*. Dicho depurador es una herramienta para analizar la ejecución del código OpenCL en la unidad correspondiente del simulador. Mediante una interfaz gráfica, actúa como una ayuda visual en la prueba del código y la conceptualización de la arquitectura del propio simulador.

En sus últimas versiones, Multi2Sim usa el formato INI para todos sus archivos de entrada y salida. Un fichero INI es un archivo de texto plano usado para almacenar la información de configuración e informes estadísticos de los programas.

Finalmente, Multi2Sim se ha adaptado para proporcionar las estadísticas que requiere McPAT como archivos de entrada. McPAT es un entorno de modelado que integra consumo, área y tiempo y que soporta la exploración exhaustiva del espacio de diseño para procesadores de muchos núcleos.

III. Metodología propuesta

El enfoque metodológico propuesto consta de cuatro etapas con un grado de dificultad creciente. El objetivo es facilitar a los alumnos interesados que puedan conseguir unos conocimientos sólidos sobre la arquitectura de los procesadores actuales llevando a cabo una interacción progresiva con el simulador. Las distintas etapas definidas son:

Modificar parámetros de simulación: Debido a la dificultad del código y con el fin de entender como los distintos parámetros afectan a las prestaciones (por ejemplo, el predictor de saltos, el ancho de lanzamiento, o el ancho del enlace de la red) en una primera etapa el alumno debe modificar el valor de algunos parámetros clave del sistema estudiados previamente de forma teórica, sacar resultados y realizar un análisis de la influencia de los mismos sobre las prestaciones del sistema. Esta primera etapa debe hacerse de manera muy guíada por lo que se propone la realización de la misma en prácticas de laboratorio.

Modificar partes del código simples: Una vez el alumno sabe lanzar simulaciones modificando valores de parámetros y es capaz de interpretar los resultados obtenidos, debe estar en disposición de modificar parte del código muy específicas y acotadas como puede ser el predictor de saltos o el mecanismo de prebúsqueda. A su vez esto debe hacerse de manera muy guíada, por ejemplo, darle un mecanismo de prebúsqueda como base, para que se lo estudie y lo modifique para obtener otro distinto. Esta etapa requiere una parte de estudio por parte del alumno. Dada la complejidad, debe prepararse una memoria muy detallada con el fin de facilitar la labor del alumno. Por otra parte, al finalizar la etapa, el alumno está preparado para enfrentarse a mayores retos con el simulador. En la metodología propuesta esta etapa se puede llevar a cabo de manera fructífera mediante trabajos de asignatura.

Implementar funcionalidades completas: Aquellos alumnos que finalicen con éxito la segunda etapa, tendrán probablemente mayores inquietudes y desearán enfrentarse a mayores retos ya que esto les permitirá conocer el funcionamiento interno de los procesadores con mucho más detalle. Esta etapa se puede implementar mediante los Proyectos Fin de Carrera (PFC) o Tesis de Máster (TM). Los resultados alcanzados en los PFCs o en las TMs por algunos de nuestros alumnos han sido espectaculares y han llegado a publicarse en congresos internacionales de primer nivel como

el IPDPS o el Euro-Par.

Autonomía completa: Aquellos alumnos que finalizan las 3 etapas anteriores con éxito se encontrarán en una posición de privilegio para empezar la tesis doctoral, ya que llevarán al menos un año de ventaja o incluso varios, sobre aquellos que no hayan empezado con esta metodología desde el principio.

IV. Caso de estudio

Este trabajo se centra en la aplicación de la metodología en las dos primeras etapas. La metodología se ha llevado a cabo con éxito durante el presente curso académico en las asignaturas de Arquitecturas Avanzadas (AAV), asignatura de segundo ciclo y máster, y Redes en Chip (REC), asignatura de máster, tanto en el ciclo de Ingeniería en Informática como en el máster en Ingeniería de Computadores del Departamento de Informática de Sistemas y Computación de la Universitat Politècnica de València, llevando a cabo un acercamiento gradual al simulador Multi2Sim por parte del alumnado.

A. Elección del trabajo y etapas de aprendizaje

Es importante que el trabajo a realizar sea sugestivo y motive al alumno a su realización. Con este propósito hemos elegido como sistema base un chip multiprocesador con ocho núcleos conectados mediante una malla. El alumno deberá ir alcanzando etapas de aprendizaje de manera progresiva, y cada una de ellas debe motivarle para alcanzar la siguiente. Hemos propuesto las siguientes cuatro etapas.

- Modelado del sistema.
- Ejecución de cargas multiprogramadas.
- Ejecución de cargas cargas paralelas.
- Implementación mecanismos de prebúsqueda.

B. Modelado del sistema base

El estudiante debe adquirir únicamente el conocimiento del simulador que le haga falta según la etapa de aprendizaje que desee alcanzar, no es aconsejable hacer un tutorial detallado del simulador antes de empezar el trabajo, ya que esto desanimaría casi con toda seguridad al estudiante. Por ello, hemos optado por la metodología "aprender haciendo".

Para poder llevar a cabo dicha labor de acercamiento, el profesorado ha preparado como primer paso para el estudiante una guía detallada para el uso del simulador. En ella, se explican todas los elementos que es capaz de modelar el simulador: core, cachés (privadas o compartidas) y red de interconexión. Para cada uno de estos elementos, se explica su funcionamiento y papel en el sistema y se enumeran cuáles son los principales parámetros de configuración en el simulador y las consecuencias de cambiarlos. Por ejemplo, en el caso del core se le explica cómo cambiar el predictor de saltos o el ancho de lanzamiento del mismo, en el caso de una caché se explica cómo cambiar el tamaño de bloque

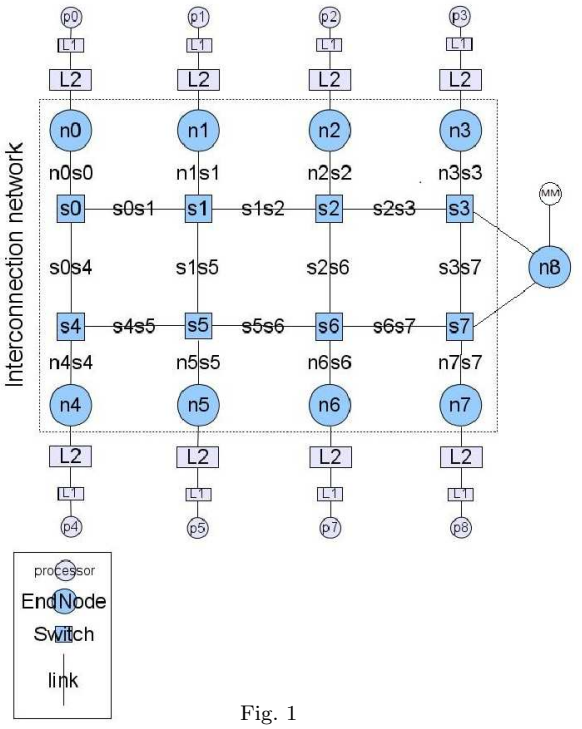

Fig. 1
Sistema inicial propuesto.

o la latencia de acceso, y en el caso de una red de interconexión se le explica cómo cambiar el ancho de banda de los canales o el tamaño de los buffers de los conmutadores. Además, se le explica en el caso de las cachés cómo modelar un sistema de memoria completo, conectando las cachés de los diferentes niveles de la jerarquía de memoria hasta llegar a la memoria principal. De forma similar, se le explica al alumno como definir una red de interconexión dentro del chip que interconecte las diferentes cachés.

Una vez explicado cómo llevar a cabo la configuración de todos los elementos del simulador, se le explica al alumno el sistema que debe modelar. Para comenzar hemos elegido un sistema CMP de 8 cores con cachés de nivel 1 y nivel 2 privadas, interconectadas por una red en topología de malla (véase Fig. 1). Indicando a su vez la configuración del core, de las cachés de los diferentes niveles y de la red, es decir, proporcionando al alumno los ficheros de configuración que modelan el sistema inicial.

```
[ Network.mesh ]
  DefaultInputBufferSize = 128
  DefaultOutputBufferSize = 128
  DefaultBandwidth = 64

;;;;;;;;;; NODES ;;;;;;;;;;
[ Network.mesh.Node.n0 ]
Type = EndNode
...
;;;;;;;; SWITCHES ;;;;;;;;
[ Network.mesh.Node.s0 ]
Type = Switch
...
;;;;;;;;;; LINKS ;;;;;;;;;;
[ Network.mesh.Link.s0s1 ]
  Source = s0
  Dest = s1
  Type = Bidirectional

[ Network.mesh.Link.s0s4 ]
  Source = s0
  Dest = s4
  Type = Bidirectional

[ Network.mesh.Link.n0s0 ]
  Source = n0
  Dest = s0
  Type = Bidirectional
...
;;;;;;;;;; ROUTING TABLES ;;;;;;;;;;
[ Network.mesh.Routes ]
  ...
  s0.to.n1 = s1
```

```
  s0.to.n2 = s1
  s0.to.n3 = s1
  s0.to.n4 = s4
  s0.to.n5 = s1
  s0.to.n6 = s1
  s0.to.n7 = s1
  s0.to.n8 = s1
  ...
```

Ejemplo 1

Archivo de configuración de la red de interconexión.

En el Ejemplo 1 se muestra parte del archivo de configuración de la red de interconexión cuyo nombre es *mesh*. Concretamente, se muestra la configuración de los parámetros globales de la red (anchos de banda y buffers), y la configuración para especificar las conexiones del nodo 0 y el switch 0, así como la tabla de encaminamiento del switch 0.

```
[ CacheGeometry l1 ]
Sets = 128
Assoc = 4
BlockSize = 64
Latency = 1

[ CacheGeometry l2 ]
Sets = 1024
Assoc = 32
BlockSize = 64
Latency = 10
...
# Level 1 Caches
[ Module dl1-0 ]
Type = Cache
Geometry = l1
LowNetwork = net-0
LowModules = l2-0

[ Module il1-0 ]
Type = Cache
Geometry = l1
LowNetwork = net-0
LowModules = l2-0
...
# Level 2 Caches
#
[ Module l2-0 ]
Type = Cache
Geometry = l2
HighNetwork = net-0
LowNetwork = mesh
LowNetworkNode = n0
LowModules = mod-mm-1
...
# Main memory
[ Module mod-mm-1 ]
Type = MainMemory
HighNetwork = mesh
HighNetworkNode = n8
Latency = 200
BlockSize = 64

# Nodes
[ Entry core-0 ]
Type = CPU
Core = 0
Thread = 0
InstModule = il1-0
DataModule = dl1-0
...
```

Ejemplo 2

Archivo de configuración de la jerarquía de memoria.

A su vez, el Ejemplo 2 muestra parte del archivo de configuración de la jerarquía de memoria. Únicamente se muestra la configuración de los tamaños de las cachés de nivel 1 y 2 y su latencia de acceso, la definición de las cachés del nodo 0, su posición en la jerarquía de memoria y conexión con su correspondiente caché de nivel 2 y la de ésta a su vez con memoria principal a través de la red *mesh* definida en el Ejemplo 1.

C. Ejecución de cargas multiprogramadas y paralelas

Es importante que el alumno aprecie como su modelo es capaz de ejecutar correctamente benchmarks que se ejecutan en los procesadores reales y son de uso generalizado por la comunidad científica. En este sentido hemos optado por el suite de benchmarks paralelos SPLASH2 [4] y la suite de benchmarks SPEC-CPU 2006 [2].

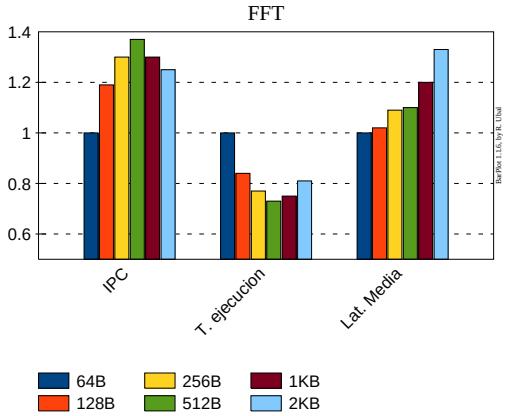

Fig. 2

Evolución de la productividad conforme se aumenta el tamaño de bloque de caché de nivel 2 para FFT. Todas las medidas son relativas a la configuración con 64B.

Para facilitar el trabajo, la guía de la prácticas proporciona los scripts de configuración para poder ejecutar las cargas correspondientes a los conjuntos de benchmarks citados y se le pide que lleve a cabo un estudio inicial de la productividad del sistema para las diferentes cargas midiendo tres indicadores: tiempo de ejecución, instrucciones por ciclo y latencia media de la red.

Una vez completada esta toma de contacto con el simulador, se indica al alumno que el objetivo final del trabajo es modelar en dicho sistema los mecanismos de prebúsqueda en memoria más comunes, sin embargo este objetivo se hará mediante tres pasos progresivos.

Primero, se le pide al alumno que varíe el tamaño de bloque de las cachés de nivel 2 de los diferentes cores haciendo que sean múltiplos del tamaño de bloque de las cachés de nivel 1, manteniendo el tamaño total de la caché de segundo nivel. De esta forma, perseguimos que el alumno vea que al aumentar el tamaño de bloque de la caché de nivel 2, se produce un efecto similar a la prebúsqueda cuando haya un fallo de memoria en el primer nivel de los cores, ya que cuando se pida un bloque desde una caché de nivel 1 debido a un fallo, si la caché de nivel 2 no lo tiene pedirá a memoria principal el bloque que ha fallado y los n sucesivos, dependiendo n de la relación de tamaños de bloque entre los niveles de caché 1 y 2. De esta forma, se le indica a los alumnos que ejecuten los diferentes benchmarks variando el tamaño de bloque de caché de nivel 2 entre 64B y 2KB, y obtengan la evolución de los indicadores de productividad. Para ver la evolución se les indica que obtengan una gráfica similar a la presentada en la Figura 2, con cada indicador normalizado con respecto a los resultados obtenidos con un tamaño de 64B y analicen las tendencias de las curvas.

D. Implementación de técnicas de prebúsqueda

Posteriormente, una vez acabado este estudio inicial pensado para que el alumno gane autonomía y confianza con el simulador, se le pide que implemente una serie de mecanismos de prebúsqueda en el

simulador. Dada la complejidad del mismo, y el estado de iniciación del alumno, hemos optado por proporcionarle ya implementado en el simulador el mecanismo de prebúsqueda más simple a modo de guía, indicando adicionalmente todos los cambios que habría que realizar en el simulador para poder llevarlo a cabo.

Concretamente, al alumno se le proporciona ya implementada la política de prebúsquda *One Block Look-Ahead* (OBL) en caso de fallo. Para ello, se ha creado en los cores una cola de prebúsquedas pendientes que se mira en la etapa *issue* de los mismos en caso de que el ancho de *issue* de las unidades de memoria no haya sido ya consumido por peticiones normales y si es posible se lanza la prebúsqueda al siguiente nivel de caché; también se ha modificado la parte de la jerarquía de memoria para añadir un disparador en caso de que se produzca un fallo, y se ha añadido código que genera una petición de prebúsqueda siguiendo la política OBL y la inserción en la cola de prebúsquedas pendientes del core correspondiente al fallo.

Una vez que el alumno se ha familiarizado con la implementación de un mecanismo básico, en este caso con del procedimiento para lanzar prebúsquedas, ya se encuentra en condiciones de realizar el trabajo de asignatura. Algunos ejemplos de trabajos de asignatura podrían ser:

Prebúsqueda n-bloques secuencial: Este trabajo consiste en la modificación de la política de prebúsqueda para que en lugar de buscar el bloque siguiente en memoria busque los n bloques siguientes, siendo n parametrizable desde los ficheros de configuración. Realizando la evaluación de prestaciones usando los mismos indicadores usados con anterioridad para un tamaño de caché dado cuando n es 1, 2, 4 y 8.

Prebúsqueda n-bloques con stride regular: Este trabajo difiere del primero en que la técnica de prebúsqueda implementada. En este caso se trata de un stride regular donde las prestaciones a alcanzar deben ser mejores, ya que el caso anterior se trata de un caso particular de éste.

Prebúsqueda secuencial para distintas topologías: Este trabajo difiere del primero en que se evalúa la prebúsqueda en distintas topologías, se centra sobretodo en el análisis del tráfico de la red.

Prebúsqueda secuencial: evaluación de la red: Este trabajo difiere del primero en que los alumnos deben estudiar cómo los parámetros de la red afectan a las prestaciones: ancho del enlace, número de buffers de en el conmutador, etc.

Tras cursar las asignaturas, algunos alumnos se sienten realmente motivados. A aquellos interesados en la temática, que se sienten capacitados y quieren llevar a cabo un PFC o TM con el simulador se les propone ampliar el trabajo de asignatura para realizar el PFC o TM.

Algunos ejemplos de trabajos de asignatura podrían ser:

Prebúsqueda adaptativa: El proyecto consiste en, basándose en el trabajo de asignatura, implementar una política de prebúsqueda adaptativa que antes de lanzar las diferentes peticiones de prebúsqueda compruebe el estado de la red, y adapte su inyección en la misma a las condiciones de prestaciones de ésta.

Asignación de cargas a cores: Se trata de estudiar políticas de asignación de cargas a núcleos de cara a maximizar las prestaciones del sistema.

V. Conclusiones

En este artículo se ha presentado la metodología puesta en marcha durante este curso en la Universitat Politècnica de València para la enseñanza práctica de la Arquitectura de Computadores con un enfoque realista y actualizado. La metodología se basa en el uso de un simulador muy completo que permite simular los aspectos más interesantes de los procesadores actuales. Debido a la complejidad del simulador, éste se utiliza en distintas asignaturas y en ellas el alumno va adquiriendo de forma progresiva un conocimiento y manejo más profundos del simulador, lo que le permite llegar a un aprendizaje realista de los procesadores actuales. Los resultados nos han demostrado que el alumno en un tiempo razonable es capaz de manejarse con el simulador y obtener resultados muy interesantes que han permitido ser publicados en congresos de relevancia internacional.

Por otra parte, desde el punto de vista del profesor, una vez diseñada la metodología y montado el entorno, hemos construido una plataforma de futuro, que cambiando el tipo de prebúsqueda, topología de red, etc. se pueden obtener una gran cantidad de trabajos para el estudio de procesadores realistas, muchos de los cuáles llegan realmente a motivar tanto al alumno como al profesor.

Agradecimientos

El presente trabajo ha sido financiado por el proyecto CICYT TIN2009-14475-C04.

Referencias

[1] Todd Austin, Eric Larson and Dan Ernst *SimpleScalar: An Infrastructure for Computer System Modeling.* IEEE Computer vol. 35(2), 2002.

[2] Cloyce D. Spradling, *SPEC CPU2006 Benchmark Tools.* SIGARCH Computer Architecture News vol. 35(1), 2007.

[3] Rafael Ubal, Julio Sahuquillo, Salvador Petit and Pedro Lopez. *Multi2Sim: A Simulation Framework to Evaluate Multicore-Multithreaded Processors.* 19th International Symposium on Computer Architecture and High Performance Computing, pages 62–68, 2007.

[4] S. Woo, M. Ohara, E. Torrie, J. Singh and A. Gupta. *The Splash-2 programs: Characterization and methodological considerations.* In 22th International Symposium on Computer Architecture (ISCA), pages 24–36, 1995.

[5] The Multi2Sim Simulation Framework Website, `http://www.multi2sim.org`