Práctica 7: *Introducción al simulador de multiprocesadores LIMES*

1. Objetivos

El objetivo de la presente práctica es aprender a utilizar el simulador de multiprocesadores Limes y realizar un estudio de los principales protocolos de coherencia de cachés basadas en bus.

Gracias al simulador Limes se pueden calcular los tiempos de ejecución de los programas suponiendo diversas configuraciones y tipos de caché. Se pueden realizar tablas y gráficas que muestren el aumento de rendimiento (*speed-up*) y el tiempo de ejecución en función del número de procesadores para cada uno de los protocolos de caché implementados (MESI, Berkeley y Dragon).

2. Desarrollo

El trabajo de laboratorio consistirá en probar el simulador con una aplicación ya realizada (la FFT al menos) y estudiar cómo afecta al rendimiento la utilización de diferente número de procesadores o la utilización de cache de diferente tamaño.

2.1 El simulador Limes

Este simulador consiste en un entorno que emula un sistema multiprocesador con memoria centralizada. Esto significa que los sistemas que se pueden simular son sistemas basados en un único bus compartido por todos los procesadores. Al mismo tiempo, y dado que sólo existe un bus, los protocolos de caché implementados son de sondeo (*snoopy*).

Este simulador se ejecuta bajo Linux y consiste en un árbol de directorios donde se encuentran las fuentes del simulador. Cada vez que se realiza una simulación hay que compilar las fuentes puesto que los diferentes parámetros de simulación se encuentran en las propias fuentes.

2.2 Pasos a seguir para realizar una simulación

Como las fuentes son el propio simulador y las vamos a modificar, es necesario que cada usuario tenga su propio simulador sobre el que realizar los cambios. Instalar el simulador es muy sencillo, basta con conectarse a la página http://www.uv.es/varnau/limes/limes.htm y seguir las instrucciones que se indican. En la página web se indican las instrucciones para la instalación completa de Limes, pero como muchas de estas operaciones ya se han realizado en las máquinas de los laboratorios, sólo es necesario bajarse el fichero con las fuentes del Limes (*limeslabac-v1.1.tgz*) descomprimirlo tal como se indica y exportar la variable *LIMESDIR* en el fichero de arranque de la *shell* de Linux. Esta operación habrá que hacerla una sola vez; cuando ya lo tengamos en nuestro directorio no será necesario volverlo a instalar (salvo que se hagan modificaciones tan grandes que se requiera volverlo a instalar todo de nuevo). La máquina virtual que contiene el simulador es: **disdo4limes**.

Suponiendo que ya se ha completado la tarea anterior, realizar la simulación de una aplicación de las que vienen con las fuentes (FFT) sólo requiere los siguientes pasos:

1. En primer lugar, hay que ir al directorio donde se encuentra la aplicación que queremos ejecutar en el multiprocesador virtual. Por ejemplo, se va a utilizar la aplicación de SPLASH-2 que realiza la transformada rápida de Fourier (FFT) para ello se hace: > *cd $LIMESDIR/applications/fft*.
2. A continuación, hay que modificar el fichero *makefile* para cambiar los parámetros que se precisen. En concreto interesan los parámetros `SIM` y `SIMHOME`. Con el parámetro `SIM` se especifica el protocolo de caché (en nuestro caso usaremos *ber* para Berkeley, *dra* para Dragon y *MESI* para MESI) y con `SIMHOME` se dice dónde se encuentra, que para los todos los protocolos es en el directorio *snoopy* salvo para el MESI que se encuentra en *snoopy/MESI*. **Por defecto ambos parámetros se encuentran comentados por lo que será necesario descomentarlos.**
3. **Antes de realizar una compilación hay que limpiar** siempre los resultados de cualquier simulación anterior, esto se hace con: > *make totalclean*. Esta operación se realizará siempre antes de hacer *make*.

4. Una vez modificado el *makefile* hay que generar el simulador para esta aplicación, para ello haremos simplemente *make*. Con esto se genera el fichero *FFT* que es el ejecutable. Este ejecutable admite varios parámetros que podemos conocer haciendo *FFT –h*. La opción *–p* nos permite especificar el número de procesadores que ejecutan la aplicación; por ejemplo, *FFT –p8* ejecuta la aplicación *FFT* como si tuviera 8 procesadores.
5. Una vez ejecutada la aplicación nos aparecerá en pantalla información sobre el tiempo consumido por los procesos en realizar sus funciones. Los parámetros que nos interesan son dos: *Total time with initialization* (tiempo total con inicialización) y *Total time without initialization* (tiempo total sin inicialización). El tiempo total con inicialización es el tiempo total invertido en realizar los cálculos incluyendo el reparto de datos entre los procesadores, etc., mientras que el tiempo sin contar la inicialización es sólo el tiempo que se ha consumido en realizar los cálculos. En general **utilizaremos el tiempo global (contando todo)** para comparar unas máquinas y otras.

3. Trabajo a realizar

3.1 Análisis de la aplicación FFT

Hay que simular la aplicación FFT que viene con el simulador y realizar un estudio tanto de los tiempos de ejecución como del aumento del rendimiento (*speed-up*) para los protocolos Berkeley, MESI y Dragon. Todo esto en función del número de procesadores. Para realizar este estudio se deberán realizar múltiples simulaciones modificando los parámetros convenientemente.

Los resultados, como no se pueden realizar gráficas a mano o imprimir las generadas por ordenador, se darán en formato de tablas. Se estudiarán los tres protocolos y se verá cómo afecta el número de procesadores utilizados al tiempo de ejecución. Las tablas en concreto mostrarán:

1. Tiempo de ejecución (contando la inicialización) en función del número de procesadores.
2. Aumento del rendimiento en función del número de procesadores.

El número de procesadores se irá variando en potencias de dos empezando en 1 y acabando en 32, es decir, hay que repetir cada simulador con 1, 2, 4, 8, 16 y 32 procesadores.

Se recuerda que el aumento del rendimiento (*speed-up*) para *n* procesadores es el tiempo de ejecución de un procesador dividido entre el tiempo de ejecución con *n* procesadores, es decir *S(n)=T(1)/T(n).*

3.2 Análisis del efecto del tamaño de la cache.

Es muy interesante que el estudiante repita este estudio utilizando un tamaño de caché diferente. Por ejemplo, la opción:

```
-- -dcache_size=X
```

permite definir una cache con un tamaño `X` expresado en Kbytes. El valor por defecto son 8 K. Se pueden probar otros tamaños desde 1 K hasta que no se note variación en el rendimiento (varios cientos de Kbytes) subiendo en potencias de dos.

4. Conclusiones

Con el estudio de las tablas y si se dispone de tiempo de las gráficas que generan, el alumno debe ser capaz de sacar conclusiones acerca de la escalabilidad del sistema, además de averiguar qué protocolos son más escalables o más rápidos.

Se dispondrá de una explicación en PPT sobre el trabajo a desarrollar en el laboratorio colgada en la WEB de la asignatura.

Todos estos conceptos serán evaluados mediante un examen de laboratorio que se entregará al inicio de la sesión de laboratorio, una vez instalado el simulador y comprendido su funcionamiento.

5. Examen.

El examen contendrá 4 preguntas, de las cuales la primera se muestra a continuación:

P1) Ejecutar el simulador LIMES con la aplicación FFT, protocolo MESI y con las siguientes características:

a) $\mathbf{2^{12}}$ datos de entrada para la FFT.

b) Variar el número de procesadores desde 1 hasta 64 {1, 2, 4, 8, 16, 32, 64}.

c) Completar las tablas y gráficas que se muestran a continuación utilizando solo **T1**, donde T1 = "*Total time with initialization*" y T2 = "*Total time without initialization*".

Práctica 7

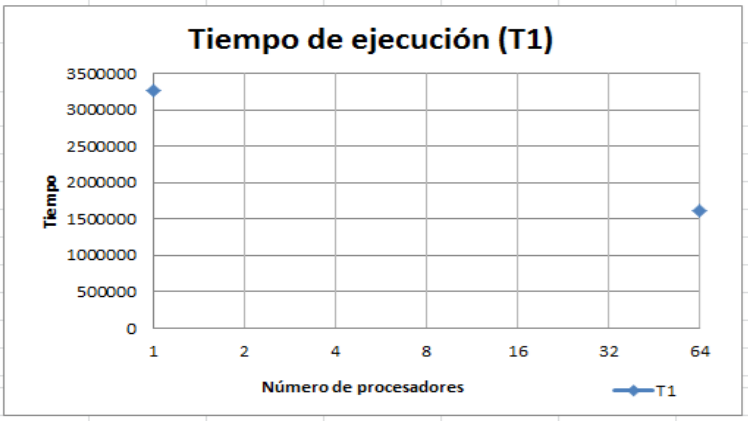
Protocolo MESI

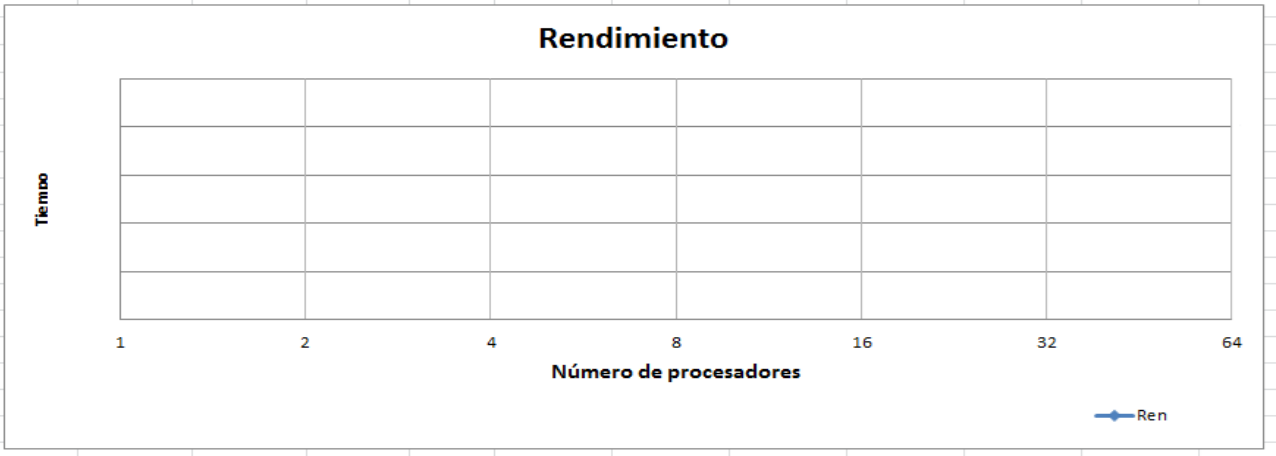
```
>./FFT  -m12  -p...
```

P	T1	T2	Ren	Ren/P
1	3272325	2742985	**1,000**	**1,000**
2				
4				
8				
16				
32				
64	1618182	936267	**2,022**	**0,032**

T1 = Total time with initialization

T2 = Total time without initialization





d) Para saber a partir de qué valor de P (número de procesadores) es aconsejable no utilizar más procesadores, se calcula la eficiencia del sistema dividiendo el rendimiento obtenido por el número de procesadores, de esta forma tenemos una nueva gráfica: