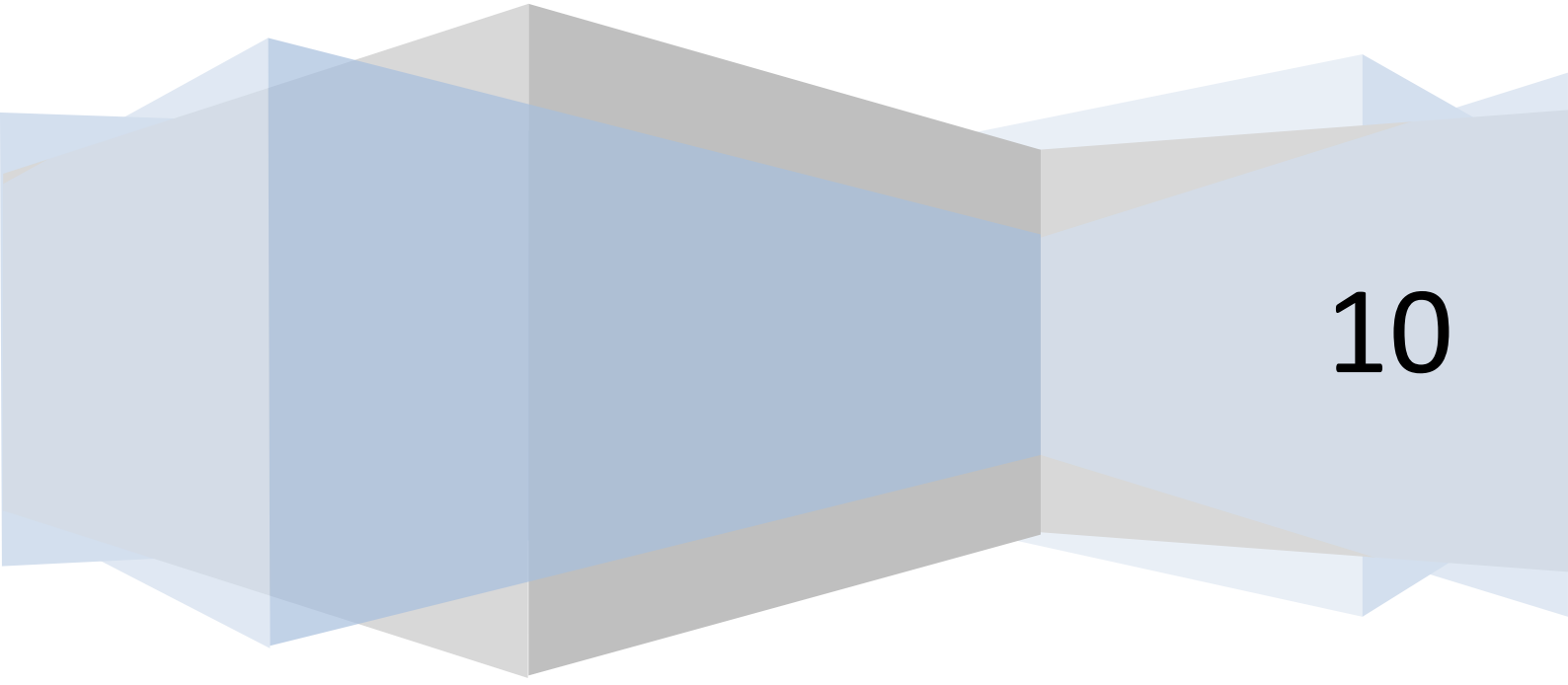
LAB5

Paralelismo a Nivel de Thread

Laboratorio de Arquitectura e Ingeniería de Computadores

Valentin Puente



10

Revision 1.1: 26/04/2010

1 INTRODUCCIÓN Y OBJETIVOS

El objetivo fundamental de esta práctica es, a partir de los resultados obtenidos con la infraestructura de simulación basada en Simics que estamos empleando, recolectar estadísticas y realizar recomendaciones en base a los resultados para una arquitectura basada en Sun UltraSparc T1 (Niagara).

Igual que en prácticas precedentes la practica tiene dos partes diferenciadas, siendo la primera imprescindible su realización y la segunda opcional. La primera parte contribuirá a un 50% de la nota de la práctica y la segunda en otro 50%. La parte dirigida es estrictamente personal mientras que la opcional puede ser llevada a cabo por grupos de hasta 2 personas.

1.1 EL TARGET

En esta práctica, continuaremos dando uso al target preparado en la segunda parte de la práctica anterior. El target será **niagara-simple** corriendo Solaris. Este procesador usa el ISA SPARCv9. Tiene hasta 8 cores cada uno de ellos con soporte hardware para la ejecución de 4 threads simultáneos (4 contextos hardware). Este procesador emplea FGMT, lo que implica que cada core conmuta entre cada uno de los 4 posibles threads ciclo a ciclo. Las configuraciones disponibles en simics para este procesador son 1 core x 1 contexto, 2 cores x 1 contexto y 8 cores x 4 contextos. Solaris considera cada contexto como una CPU separada.

La configuración será mejorada sobre la usada en la práctica anterior con la inclusión de un nuevo nivel de cache. En esta práctica, cada core tendrá su propia cache L1 unificada. Lógicamente cada uno de los 4 threads que se ejecuten en el mismo core compartirán esa cache. Todas las caches L1 son *write-thorugh* y son respaldadas por un segundo nivel de cache que es compartido por todos los cores. La coherencia de datos entre las diferentes caches del sistema es mantenida mediante un protocolo MESI basado en bus, similar al descrito en clase.

1.2 LA CARGA MULTI-THREAD

El benchmark que emplearemos en la práctica es una implementación paralela de una simulación 3D de turbulencia de plasma utilizando el método de Lattice-Boltzman. La implementación realizada usa **pthreads** como librería de paralelización. El programa con los siguientes argumentos de entrada:

```
lbmd_solaris [tamaño_en_X] [tamaño_en_Y] [tamaño_en_Z]
[número_de_threads] [nivel_de_optimización_del_compilador (0-2)]
[tamaño_de_vector (1-128)] [cantidad_de_unrolling (1-8)] [ILP (1-
8)]
```

Los tres primeros parámetros especifican el tamaño de problema, el cuarto el numero de threads en los que vamos a dividir los cálculos y el resto son parámetros de optimización que vamos a utilizar durante la ejecución. Dado que no tenemos compilador disponible en la máquina simulada, hemos optado por enlazar dentro de un único ejecutable diferentes objetos para la función que resuelve el problema. Cada objeto ha sido compilado con diferentes opciones de compilación. Estos son seleccionables a través de los 5 parámetros de configuración final.

1.3 CALIFICACIÓN

Para la parte obligatorio, deberás entregar en formato ***pdf***[1] en través de la tarea abierta en WebCT un documento que responda al final del apartado 2.2 de la práctica. El trabajo opcional se presentara en público.

2 PARTE DIRIGIDA

2.1 CONFIGURACIÓN

Antes de pasar a realizar la práctica, deberemos preparar el checkpoint base. Para ello llevaremos a cabo los siguientes pasos:

1. Iniciar Simics

```
host$ ./simics
```

2. Configurar el numero de CPUs lógicas

```
simics> $num_cpus=32
```

3. Ejecutar el script de configuración correspondiente a **niagara-simple**. El arrancar el target va a requerir una cantidad de tiempo substancial. Si estas usando una máquina virtual y el procesador de tu PC es inferior a un Core 2 Duo, te recomiendo[2] emplear el equipo del laboratorio y que preferiblemente sea de los nuevos.

[1] Cualquier documento que no se adecue a este formato (rar, doc, docx, etc…) no será evaluado

[2] En un PC poco potente puedes hacer el arranque de la máquina y llegar hasta el punto 8 antes de ir al laboratorio. En pasos sucesivos una máquina lenta no debería tardar demasiado.

```
simics> run-command-file targets/niagara-simple/niagara-simple-solaris-common.simics
```

4. Copiar el ejecutable del benchmark (`lbmhd_solaris`) en el **target**

```
target# mount /host

target# cp /host/<path_ejecutable>lbmhd_solaris ./
```

5. Crear un checkpoint. Esto requiere una cantidad substancial de espacio disponible. Verifica (`dh -h` y `quota -v`) si al menos tienes disponibles 220 MB.

6. Añadir un punto de ruptura al ejecutable con:

```
simics> run-command-file add-breakpoint.simics
```

7. Iniciar la ejecución del benchmark con:

```
target# ./lbmhd_solaris 32 32 32 32 1 8 2 1
```

8. Crear un checkpoint y salir. La configuración para el resto de la práctica esta lista.

2.2 EJECUTANDO UN CÓDIGO MULTITHREAD EN EL SUN ULTRASPARC T1

En esta sección ejecutaremos una carga multi-thread para estudiar cómo se comporta el sistema con la configuración de la jerarquía de memoria previamente descrita. Para ello haremos los siguientes pasos:

1. Iniciar simics en modo **stall**, cargando el checkpoint generado en el punto anterior. Suponiendo que LAB_AIC5 es el directorio donde has descomprimido los ficheros facilitados para la realización de esta práctica, carga la configuración de la jerarquía de memoria con:

```
simics> run-command-file add-niagara-2level-cache.simics
```

2. Ejecuta la simulación durante 1 millón de ciclos, para calendar las caches.

3. Una vez calentadas, reinicia sus estadísticas con:

```
simics> l2cache.reset-statistics

simics> l1cache0.reset-statistics
```

```
simics> l1cache1.reset-statistics
```

…

4. Continua la simulación durante al menos otros 10 millones de ciclos.
5. Examina las estadísticas para la caches L1 y L2. Date cuenta que ahora el comando de estadísticas retorna estadísticas del protocolo de coherencia. Anota los datos obtenidos.

6. Los datos obtenidos en el punto anterior corresponden a una cache L1 de 8KB y 512 KB de L2. Cambia el tamaño de las caches L1 a 64KB y repite los pasos anteriores. Para modificar el fichero de configuración se facilita un script en Python que automatiza el proceso. Cambia el número de líneas en este fichero.

7. Repite el proceso anterior pero multiplicando por 2 el tamaño de bloque en L1

¿Cómo ha cambiado el comportamiento de la cache en las tres configuraciones propuestas? ¿Y el comportamiento del protocolo de coherencia? Explicar las observaciones.

3 PARTE ABIERTA

3.1 EXPLORACIÓN DEL ESPACIO DE DISEÑO

Supón que eres el Arquitecto Jefe de la siguiente generación del Niagara. Por restricciones de coste solo dispones de 1MB de cache para incluir el chip. Modifica el script Python proporcionado para probar diferentes diseños y quedarte con el más apropiado, teniendo en cuenta que en total todas las caches pueden tener capacidad para 1MB. Puedes cambiar el número de líneas de cache, tamaño del bloque y asociatividad. No se puede tocar la política de escritura: estate seguro de que solo el nivel más bajo es write-back.

La estructura base que hemos estado manejando en los puntos anteriores (8 caches L1 privadas unificadas para datos e instrucciones y 1 cache compartida L2). Puedes quedarte con esta configuración. Si por el contrario te sientes creativo puedes añadir nuevos niveles o cambios más profundos como emplear una cache L2 también privada. En el punto **18.10** de la guía de usuario de Simics se explica como el módulo **g-cache** puede ser usado para jerarquías con varios procesadores. Recordar que es preciso determinar los tiempos de acceso cada nivel de la jerarquía con CACTI.

Evaluar cual es la eficacia de las configuraciones elegidas reportando los GOPs/seg al final de la ejecución (lo que puede llevar una cantidad considerable de tiempo) o alguna medida alternativa de rendimiento que se te ocurra (¿¿¿CPI???). Además sería interesante observar la eficacia de la jerarquía de memoria en cada caso. Argumentar por que es mejor la mejor configuración y peor la peor configuración.

3.2 PROPONER UN PROYECTO

Propón algo que consideres interesante. Cualquiera de los temas tratados en la asignatura será adecuado. También lo serán otros no directamente tratados pero relacionados. Eres libre de emplear las herramientas software que consideres oportuno. Eso sí,… antes de lanzarte a la piscina sería recomendable discutirlo con el profesor.

4 REFERENCIAS

[1].- Simics User Guide for Unix: <WebCT>.

[2].- CACTi Web interface http://quid.hpl.hp.com:9081/cacti/