Práctica 4: Explotar el potencial de las arquitecturas modernas

NOTA: Leer detenidamente el enunciado de cada ejercicio antes de proceder a su realización.

El objetivo de esta práctica es experimentar y aprender a sacar partido a las arquitecturas *multicore (o manycore)* a través del *framework* de programación OpenMP [1].

Todo el trabajo asociado a esta práctica se realizará en un entorno Linux (en particular, asumiremos que se utiliza la versión de Linux instalado en los laboratorios: Ubuntu 14.04).

MATERIAL DE PARTIDA

Junto con el enunciado de la práctica, se entrega el "`MaterialP4.zip`" que contiene los siguientes ficheros que serán referenciados a lo largo del enunciado:

- `omp1.c` – fichero con el código de ejemplo de un programa que crea hilos utilizando OpenMP. El número de hilos creados se pasa como argumento al programa.
- `omp2.c` – fichero con el código de ejemplo de un programa que muestras las diferencias en la declaración de la privacidad de variables en OpenMP.
- `pescalar_serie.c` – fichero con un código serie que realiza el producto escalar de dos vectores.
- `pescalar_par1.c` y `pescalar_par2.c` – ficheros con el código de ejemplo de programas que realizan paralelización de bucles con OpenMP
- `Makefile` – fichero makefile utilizado para compilar todos los ficheros de código fuente.

IMPORTANTE

A lo largo de este documento se hará referencia a un número P que afectará a los experimentos y resultados que se obtendrán. El valor de P que debe utilizar debe ser igual al número de pareja módulo 8 más 1 (es decir será un número entre 1 y 8).

Ejercicio 0: información sobre la topología del sistema (1 p)

En primer lugar, deberíamos de ser capaces de obtener la información relativa a la memoria caché de la máquina sobre la que estamos trabajando. Esto es sencillo en un sistema Linux y podemos hacerlo ejecutando cualquiera de los siguientes comandos:

```
> cat /proc/cpuinfo
```

```
> dmidecode
```

NOTA: este comando requiere permisos de superusuario, por lo que NO podremos utilizarlo en la instalación de los laboratorios.

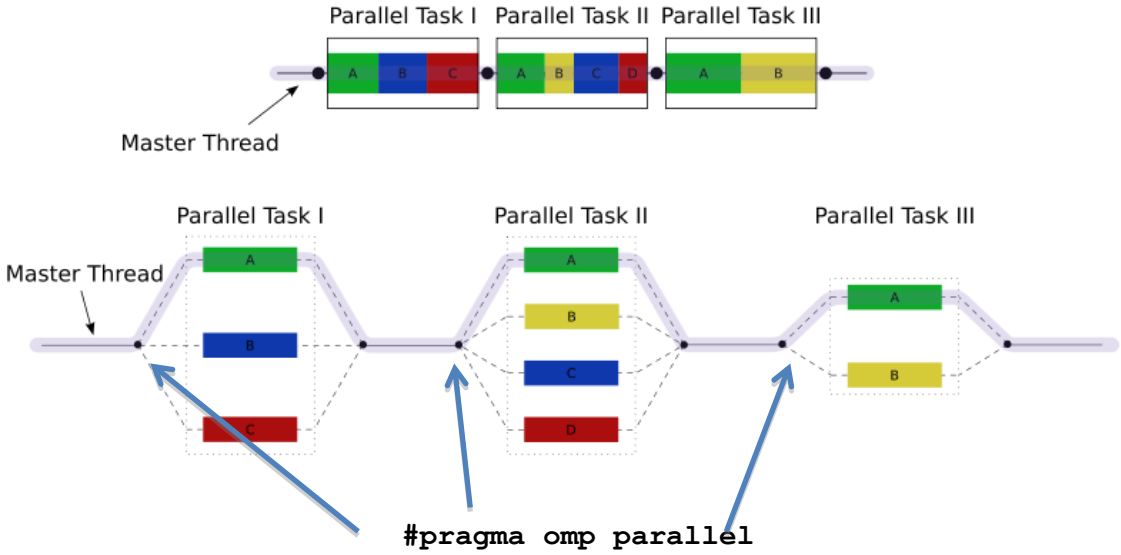
En particular, si nos fijamos en la información que nos da el fichero `cpuinfo` podemos deducir cuántos *cores* tiene nuestro equipo y si tiene o no habilitado la opción de *hyperthreading* (el parámetro `cpu_cores` indica cuántas CPUs físicas hay en equipo, y el parámetro `siblings` hace referencia al número de CPUs virtuales).

A partir de la información expuesta, indique en la memoria asociada a la práctica los datos relativos a las cantidad y frecuencia de los procesadores disponibles en los equipos del laboratorio, y a si la opción de *hyperthreading* está o no activa.

Ejercicio 1: Programas básicos de OpenMP (2 p)

En este ejercicio, vamos a realizar una primera toma de contacto con programas basados en OpenMP. Para realizar la compilación de estos programas utilizaremos como compilador `gcc`, añadiendo las banderas `-lgomp` y `-fopenmp` para indicarle que queremos utilizar las librerías y extensiones de OpenMP.

OpenMP nos permite lanzar de una forma muy sencilla un número de hilos (o *threads*) para que lleven a cabo tareas de forma paralela (cada hilo ejecutando en su CPU correspondiente). La creación de un equipo de hilos se lleva a cabo cada vez que se incluye el la directiva `#pragma omp parallel` en el programa, y la sincronización y finalización de los hilos se realiza al terminar la región paralela señalada con el `parallel`.



En el material de partida, se pide compilar y ejecutar el programa `omp1.c` para comprobar cómo se lleva a cabo el lanzamiento de equipos de hilos, y familiarizarse con las funciones `omp_get_num_threads()` y `omp_get_thread_num()` que nos permiten identificar a cada uno de los *threads* en activo.

Una característica fundamental a la hora de programa utilizando OpenMP es definir correctamente qué variables han de ser públicas (accesibles a todos los hilos, es el valor por defecto) o privadas (cada hilo accede a su propia copia). Un ejemplo de la utilización de estas características puede encontrarse en el programa `omp2.c`.

Se pide ejecutar el programa omp2 e incluir la salida en la memoria de la práctica. En base a la salida obtenida (y tras leer y comprender el código fuente que la genera) conteste a las siguientes preguntas:

1. ¿Cómo se comporta OpenMP cuando declaramos una variable privada?
2. ¿Qué ocurre con el valor de una variable privada al comenzar a ejecutarse la región paralela?
3. ¿Qué ocurre con el valor de una variable privada al finalizar la región paralela?
4. ¿Ocurre lo mismo con las variables públicas?

Ejercicio 2: Paralelizar el producto escalar (3 p)

Tomaremos como código de referencia el del producto escalar, cuya versión serie y dos versiones paralelas se entregan como material de partida de la práctica. OpenMP está especialmente pensado para realizar la paralelización del trabajo de un bucle *for* entre los hilos del equipo. Cuando se quiere repartir el trabajo de un bucle de esta forma se utiliza la cláusula `#pragma omp parallel for`. Al indicar esta cláusula, OpenMP ya hace trabajo por nosotros como por ejemplo declarar como privada la variable que se utiliza como índice del bucle.

Ejecute la versión serie y las dos versiones paralelas del producto escalar y observe el resultado numérico obtenido. Conteste en la memoria a las siguientes preguntas:

1. ¿En qué caso es correcto el resultado?
2. ¿A qué se debe esta diferencia?

Tomando ahora únicamente la ejecución de la versión serie y la versión paralela que da el resultado correcto, se pide hacer un análisis del rendimiento obtenido con 1, 2, 3 y 4 hilos paralelos en el problema del producto escalar. Analice cómo varía el rendimiento para vectores de tamaño 1000, 2000, 5000, 10000, 20000, 30000, 40000, 50000 y 100000.

La presentación de estos resultados se deberá incluir en la memoria de la práctica graficando incluyendo una gráfica con los tiempos de ejecución y otra con la aceleración obtenida.

A la luz de los resultados obtenidos, se pide contestar a las siguientes preguntas:

1. En términos del tamaño de los vectores, ¿compensa siempre lanzar hilos para realizar el trabajo en paralelo, o hay casos en los que no?
2. Si compensara siempre, ¿en qué casos no compensa y por qué?
3. ¿Se mejora siempre el rendimiento al aumentar el número de hilos a trabajar?
4. Si no fuera así, ¿a qué debe este efecto?

Ejercicio 3: Paralelizar la multiplicación de matrices (4 p)

Para este ejercicio se tomará como código de partida el de la multiplicación de matrices que cada alumno ha desarrollado a lo largo de la práctica 3 de la asignatura.

En este caso, el programa se compone de tres bucles anidados, por lo que a la hora de llevar a cabo la paralelización del trabajo a realizar es razonable preguntarse: ¿cuál de los bucles interesa paralelizar?

Se pide desarrollar y entregar (además de la versión serie de la práctica 3) tres versiones paralelas de la multiplicación de matrices paralelizando los tres bucles existentes. Las versiones entregadas han de funcionar de forma correcta.

Se pide rellenar las siguientes tablas realizando la ejecución para matrices de tamaño 1000x1000:

Tiempos de ejecución (s)

Versión\# hilos	1	2	3	4
Serie				
Paralela – bucle1				
Paralela – bucle2				
Paralela – bucle3				

***Speedup* (tomando como referencia la versión serie)**

Versión\# hilos	1	2	3	4
Serie	1			
Paralela – bucle1				
Paralela – bucle2				
Paralela – bucle3				

NOTA: se considera que el bucle 1 es el más interno, el bucle 2 es el bucle intermedio, y el bucle 3 es el bucle más externo. En el caso serie completar sólo la columna para un hilo.

A la luz de los resultados obtenidos, conteste a las siguientes preguntas:

1. ¿Cuál de las tres versiones obtiene peor rendimiento? ¿A qué se debe?
2. ¿Cuál de las tres versiones obtiene mejor rendimiento? ¿A qué se debe?

Tome tiempos de ejecución de la versión serie y de la mejor versión paralela obtenida anteriormente (la mejor combinación entre las tres versiones de código, y las 4 posibilidades para el número de hilos paralelos). Tome tiempos en un fichero y realice una gráfica de la evolución del tiempo de ejecución y el *speedup* (de la versión paralela vs serie) al ir variando el tamaño de las matrices de NxN para N entre P y 1024+P (con incrementos en N de 32). Tras incluir estas dos gráficas en la memoria, incluya un párrafo describiendo y justificando el comportamiento observado en las mismas.

MATERIAL A ENTREGAR

Memoria con las respuestas a la preguntas a lo largo del enunciado.
En subdirectorios independientes, los ficheros con los datos obtenidos en los experimentos de los ejercicios que así lo indiquen, y los scripts utilizados para la obtención de estos datos (si se ha usado alguno).
Códigos fuente desarrollados para la resolución del ejercicio 3.

La entrega se realizará a través de moodle, con fecha tope el día antes de la realización del examen correspondiente a las prácticas 3 y 4.

MATERIAL DE REFERENCIA

[1] **OpenMP: Tutorials and technical articles** http://openmp.org/wp/resources/#Tutorials