UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DSIC DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Máster Universitario en Computación en la Nube y de Altas Prestaciones

Tecnología de la Programación Paralela (TPP).

Ejercicios OpenMP Sesión 1

Ejercicio 1: Paralelización de bucles en OpenMP

Dado un conjunto de vectores, en este ejercicio se va realizar el cálculo de la *media* ($\mu = \frac{1}{n}\sum_{i=1}^{n} v_i$) y la *desviación típica* ($\sqrt{\frac{1}{n}\sum_{i=1}^{n}(v_i - \mu)^2}$) para cada vector v. Para ello, se va a trabajar con el fichero fuente facilitado (`bucle.c`). Este código genera un conjunto de vectores de diferente tamaño y de valores aleatorios.

Tarea a realizar: En la zona indicada para ello, hay que incluir un código que sea capaz de calcular la *media* y la *desviación típica* de todos los vectores generados. Observad que cada vector esta "apuntado" por un puntero del vector `M`, es decir, `M[v]` es el vector v, para $v = 1, \ldots, n\_vectores$, donde $n\_vectores$ es el número de vectores. Para almacenar la *media* y la *desviación típica* de cada uno de los vectores generados aleatoriamente se han declarado dos vectores dinámicos, `media` y `desvt`, respectivamente, de manera que los valores de la *media* y la *desviación típica* del vector `v` se almacenarán en `media[v]` y `desvt[v]`, respectivamente. El código secuencial debería estar basado en **un solo bucle** que recorra todos los vectores y, para cada vector, calcule su *media* y su *desviación típica*.

El programa guarda los valores calculados en un fichero. De esta manera se puede comprobar si el resultado es correcto entre diferentes ejecuciones mediante el comando Linux `cmp`.

Una vez realizado el código secuencial hay que paralelizarlo. Esto lo realizaremos siguiendo un "diseño basado en hilos", es decir, mediante paralelización de bucles.

Se puede utilizar la máquina que se desee pero la recomendación es utilizar `knights`. La compilación se puede realizar así:

```
$ gcc -o bucle bucle.c ctimer.c -lm
```

El programa y su paralelización han de ser estudiadas. Aunque no es necesario presentar una memoria para estos ejercicios, sí es necesario ser capaz de estudiar la paralelización. Se recomienda trabajar con tamaños, por ejemplo, de unos 40000 vectores de un tamaño máximo de 40000. El estudio del algoritmo paralelo debe consistir en mostrar la evolución de los tiempos cuando el número de cores es de 1, 2, 4, 8, 16 y 24. Además, del tiempo es necesario sacar el incremento de velocidad o *speedup* y la eficiencia para cada caso. Como la paralelización se ha realizado sobre uno o varios bucles, hay que estudiar la diferencia entre una política de planificación de bucles *estática* y otra *dinámica* para todas las combinaciones de número de hilos utilizados. Se puede reducir el estudio a dos o tres tamaños de "`chunk`" en cada caso.

Ejercicio 2: El problema de la mochila

El *problema de la mochila* unidimensional o, en inglés, *the one-dimensional 0/1 knapsack problem*, se puede definir de la siguiente manera. Se dispone de una mochila con capacidad c y una serie de n objetos numerados como $1, 2, \ldots, n$. Cada objeto tiene un peso w_i y un valor p_i. Sea $v = [v_1, v_2, \ldots, v_n]$ un vector

solución en el que $v_i = 0$ si el objeto i no está presente en la mochila, o $v_i = 1$ si lo está. El objetivo es el de encontrar un subconjunto de objetos tal que

$$\sum_{i=1}^{n} w_i v_i \leq c,$$

y

$$\sum_{i=1}^{n} p_i v_i,$$

sea máximo.

Existen diferentes métodos para resolver este problema. El método que se proporciona en el fichero `knapsackBF.c` es el más simple, conocido como de *Fuerza Bruta*, y consiste en generar las 2^n posibles soluciones y elegir la mejor. El objetivo es el de paralelizar dicho código con OpenMP. Hacedlo sin cambiar código, simplemente añadiendo las directivas de OpenMP adecuadas.