Práctica Ligera con OpenMP: Multiplicación Paralela de Matrices en Máquinas de Memoria Compartida

Como indica el título, habrá que realizar una aplicación paralela que calcule el producto de dos matrices de números en coma flotante (*float*) de grandes dimensiones. La idea es ver el beneficio que se puede obtener con una aplicación paralela frente a una secuencial. Por tanto, habrá que tener un programa secuencial, con el que comparar los resultados que obtengamos, para ver que nuestro programa funciona correctamente, así como para comparar los tiempos obtenidos.

A la hora de realizar la aplicación paralela, se empleará OpenMP, por lo que la mayor parte del trabajo la realizará el compilador, quedando oculta, para el programador, la complejidad de la tarea. De hecho, la aplicación será muy sencilla.

Normalmente habrá tantos threads como CPU's tiene la máquina, ya que si ponemos menos habrá CPU's ociosas, y si ponemos más habrá cambios de contexto innecesarios. En nuestro caso, como hay cuatro núcleos por máquina, utilizaremos cuatro threads.

La forma más sencilla de medir el tiempo empleado por la aplicación es con el comando *time* de la *shell* (en /usr/bin). Para evitar medidas erróneas se aconseja repetir cada medida un par de veces para asegurarse de que está bien (si las dos medidas difieren bastante alguna está mal y hay que repetir la medida). Para evitar distorsiones provocadas por la red, los ficheros estarán locales en la máquina.

A) Programa Secuencial

Habrá que realizar un programa secuencial, que servirá de referencia a los paralelos:

```
for (i=0; i<M1; i++)
    for (j=0; j<N2; j++) {
        acum = 0;
        for (k=0; k<N1; k++) {
            acum += mat1[i][k] * mat2[k][j];
        }
        matR[i][j] = acum;
    }
```

Este programa será el que se emplee de referencia para las siguientes secciones. Está disponible en la página web, como mat_seq.c y matriz.h.

B) Programa paralelo

A continuación se debe paralelizar con OpenMP este programa, empleando cuatro *threads.* Indique claramente que variables son privadas y cuales son compartidas. Para no olvidarse ninguna se recomienda poner `default(none)`, así el compilador nos preguntará por todas las variables que no hayamos indicado. Indique el número de *threads* a emplear. Se plantearán dos versiones:

- Paralelizar el bucle más externo (*for* con índice “i”). El grano de trabajo será cada fila de MatR
- Paralelizar el bucle intermedio (*for* con índice “j”). El grano de trabajo será cada elemento de MatR

Minimice las diferencias respecto al secuencial y asegurese de que compila correctamente con un compilador “normal”, que no tuviese OpenMP. Especifique como algoritmo de planificación el estático.

Compile con el **gcc con optimización (-O3)**. Mida tiempos y compare los obtenidos con cada versión frente a la secuencial (con los mismos flags de compilación) y calcule el **speedup** obtenido. Para que los resultados sean comparables, se emplearán los siguientes tamaños de matrices: a) 200000x50 * 50x100, b) 8000x900 * 900x300, c) 800x900 * 900x3000, d) 8000x90 * 90x8000 y e) 8000x10 * 10x8000

¿Por qué funciona unas veces mejor por filas y otras veces por columnas? Comente si son necesarias regiones críticas o barreras y por qué.

Observación

Para simplificar al máximo aspectos no relativos a la práctica, como la gestión de memoria, se recomienda emplear matrices estáticas, de forma que no habría que asignarles memoria al inicio de la ejecución ni liberarlas al final de la aplicación.

La ejecución de los distintos casos nos dará una tabla de tiempos para cada apartado. Deberá comentar y razonar por qué se dan estas medidas o resultados. A partir de estas explicaciones deberá comentar las conclusiones que saca sobre los resultados obtenidos.

En la memoria se incluirá el código, así como las medidas obtenidas en cada caso. Además, se pondrán las explicaciones a dichas medidas, las conclusiones que se pueden obtener a partir de estas medidas y los comentarios personales sobre la práctica realizada.

Observación: Compilación

A la hora de compilar no hay que tener especial cuidado, es sencillo, basta con:

- Incluir el fichero de cabecera **`<omp.h>`**
- Y compilar con: **`gcc –fopenmp fich.c –o fich`**