Práctica Nro. 3

Programación con OpenMP

Se seleccionará un ejercicio que los alumnos deberán entregar en grupos de dos personas.

Pautas:

Compilar en Linux gcc:

gcc -fopenmp –o salidaEjecutable archivoFuente

Ejecutar:

Ejercicio 1:

./ejercicio1 N cantidadDeThreads

Ejercicio 2:

Compilar utilizando la opción –lm:

gcc -lm -fopenmp –o salidaEjecutable archivoFuente

./ejercicio2 N cantidadDeThreads

Ejercicio 3:

./matrices N cantidadDeThreads

Ejercicio 4:

./traspuesta N cantidadDeThreads

Ejercicio 5:

./mxm N cantidadDeThreads

1. El programa ejercicio1.c inicializa una matriz de N*N de la siguiente manera: A[i,j]=i*j, para todo i,j=0..N-1. Compilar y ejecutar. ¿Qué problemas tiene el programa? Corregirlo.

2. Analizar y compilar el programa ejercicio2.c. Ejecutar varias veces y comparar los resultados de salida para diferente número de threads ¿Cuál es el problema? Corregirlo.

3. El programa matrices.c realiza la multiplicación de 2 matrices cuadradas de N*N (C=AxB).
   Utilizando *pragma parallel omp for* Paralelizarlo de dos formas:
   a. Repartiendo entre los threads el cálculo de las filas de C. *Es decir, repartiendo el trabajo del primer for.*
   b. Repartiendo el cálculo de las columnas de cada fila de C. *Es decir, repartiendo el trabajo del segundo for.*

   Comparar los tiempos de ambas soluciones variando el número de threads.

4. El programa traspuesta.c calcula la transpuesta de una matriz triangular de N*N. Compilar y ejecutar para 4 threads comparándolo con el algoritmo secuencial.
   El programa tiene un problema, describir de que problema se trata. ¿Qué cláusula usaría para corregir el problema? Describir brevemente la cláusula OpenMP que resuelve el problema y las opciones que tiene. Corregir y ejecutar de nuevo comparando con los resultados anteriores.

5. El programa mxm.c realiza 2 multiplicaciones de matrices de M*M (D=A*B y E=C*B). Paralelizar utilizando secciones de forma que cada una de las multiplicaciones se realice en una sección y almacenar el código paralelo como mxmSections.c. Compilar y ejecutar sobre diferente número de threads.

   Probar con 2 threads. Luego con 4 threads ¿Se Consigue mayor speedup al incrementar la cantidad de threads? ¿Por qué?