Práctica Nro. 3

Programación con OpenMP

Información útil para compilar y ejecutar:

- Para compilar en Linux con gcc+OpenMP: *gcc -fopenmp –o salidaEjecutable archivoFuente*
- Para ejecutar: *./salidaEjecutable arg1 arg2 … argN*

Pautas generales

- Para obtener el tiempo de ejecución de todos los algoritmos se debe utilizar la función provista por la cátedra (*dwalltime*).
- Por convención sólo deberá tomarse el tiempo de ejecución del procesamiento de datos. Esto significa excluir del tiempo de ejecución:
  - *Reserva y liberación de memoria.*
  - *Inicialización de estructuras de datos.*
  - *Impresión y verificación de resultados.*
  - *Impresión en pantalla (printf)*
- Las pruebas deben realizarse de forma aislada a la ejecución de otras aplicaciones. Se debe ejecutar desde consola, sin otras aplicaciones ejecutándose al mismo tiempo.
- Para todos los ejercicios se debe calcular el speedup y la eficiencia del algoritmo paralelo respecto al secuencial.

Ejercicios

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

2. Analice y compile el programa *ejercicio2.c*. Ejecute varias veces y compare los resultados de salida para diferente número de threads. ¿Cuál es el problema? ¿Se le ocurre una solución?
   Nota: al compilar, agregue el flag *-lm.*

3. El programa *matrices.c* realiza la multiplicación de 2 matrices cuadradas de NxN (C=AxB).
   Utilizando la directiva *parallel for* paralelice 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.*

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

4. El programa *traspuesta.c* calcula la transpuesta de una matriz triangular de NxN. Compile y ejecute para 4 threads comparándolo con el algoritmo secuencial.
   Si bien el programa computa correctamente la transpuesta, éste tiene un problema desde el punto de vista del rendimiento. Analice las salidas y describa de qué problema se trata.
   ¿Qué cláusula se debe usar para corregir el problema? Describa brevemente la cláusula OpenMP que resuelve el problema y las opciones que tiene. Corrija el algoritmo y ejecute de nuevo comparando con los resultados anteriores.

5. El programa *mxm.c* realiza 2 multiplicaciones de matrices de MxM (D=AxB y E=CxB). Paralelizar utilizando *sections* de forma que cada una de las multiplicaciones se realice en una sección y almacenar el código paralelo como *mxmSections.c*. Compile y ejecute con 2 threads y luego con 4 threads, ¿se consigue mayor speedup al incrementar la cantidad de threads? ¿Por qué?