Speed Challenge #2

En el Campus Virtual (y en el directorio compartido ***/home/alumnos/ac/alumnos/CH2***) tenéis el archivo fuente del segundo reto (`CH2.c`) y un script (`CH2-SLURM.txt`) para lanzar un trabajo de compilación y ejecución del programa. Se trata de un código que realiza operaciones sintéticas, sin ninguna aplicación práctica, pero que sirve para retaros a transformarlo en el código más rápido que sea posible. Si habéis intentado el reto 1, veréis que el programa propuesto resuelve el mismo problema, pero usando estrategias algorítmicas algo diferentes. Las pruebas de "velocidad" las haréis en el computador **aolin23**, que es un ordenador que está siempre disponible y en el que podéis asegurar que vuestro código se ejecute de forma exclusiva (sin que se ejecuten otros programas de forma simultánea al vuestro). Tenéis la descripción de las características de este computador también en el Campus Virtual.

Reglas del Reto

Entrega y Evaluación

Debéis entregar a través del Campus Virtual el nuevo archivo **CH2.c** y un comentario adicional de texto donde se indique: (1) el nombre de las alumnas o alumnos que forman parte del grupo (máximo dos), (2) las órdenes exactas necesarias para compilar el programa, y que en las pruebas incorporaremos al script `CH2-SLURM.txt`, y (3) el tiempo obtenido en las pruebas realizadas con los valores de entrada por defecto. Si no se proporcionan estos tres datos, incluido el tiempo de ejecución, la entrega no será válida y no será evaluada. Se pueden usar los compiladores disponibles en el laboratorio, con las opciones de compilación que se desee. Para usar estos compiladores tenéis que modificar el script de SLURM que os proporcionamos. Como tiempo de ejecución se usará el que proporciona **perf** al ejecutar el programa en el procesador **aolin23.uab.es** en modo exclusivo. Se considerará que un código es más rápido que otro si el tiempo de ejecución es al menos un 1% inferior.

Reglas de Modificación del Código

El archivo **CH2.c** se puede modificar tanto como se desee, incluyendo código en lenguaje ensamblador (aunque, por supuesto, no es nada recomendable el uso de lenguaje ensamblador, que reflejaría el fracaso en ser capaz de dominar al compilador para que trabaje él por nosotros). En esta entrega sí que se permite paralelizar el programa en threads para usar los múltiples núcleos de cómputo del procesador, pero no se permite usar la GPU del computador. Se permite cambiar las estructuras de datos, expresiones aritméticas, orden de ejecución de sentencias, modificar funciones y usar funciones *intrinsic*. No se pueden incluir llamadas a funciones complejas de bibliotecas que implementen algoritmos, tanto las del standard C, como otras más especializadas, como **STL** o **MKL**. Por ejemplo, no se pueden usar llamadas a la función **qsort**() o a otra función de ordenación que se encuentre en una biblioteca. Sí se pueden usar funciones básicas, como **malloc**, **free**, **memcopy** …

El código entregado debe ser **funcionalmente equivalente** al original para cualquier combinación de datos que se use como entrada, asumiendo que siempre D >= 2000, N >= 10, Iter >= 50). La salida funcional del programa (la parte que no hace referencia al rendimiento de la ejecución) debe producir resultados idénticos a los de la versión original (se pueden obtener las diferencias con el comando **diff**). Como las operaciones son todas con datos enteros, no debe haber ninguna diferencia en los resultados. Los profesores comprobaremos que el resultado funcional del programa que entreguéis es idéntico al de la versión original para un ejemplo de datos de entrada diferente.

Compilar y Ejecutar el programa:

Usaréis un script (`CH2-SLURM.txt`) para lanzar un trabajo de compilación y ejecución de vuestro programa al computador **aolin23** usando el gestor de colas **SLURM**. Para ello tenéis que conectaros remotamente al computador **aolin-login** o **aolin22,** copiar los archivos fuente del reto y el script en un directorio, y desde ese mismo directorio teclear:

```
$ sbatch -o OUT.txt CH2-SLURM.txt
```

A continuación, usad el comando `squeue` para visualizar el estado de la ejecución del trabajo. Poco tiempo después de que el trabajo haya finalizado os encontraréis el fichero `OUT.txt` en el directorio actual, que contiene la salida de la ejecución, incluyendo el tiempo total de ejecución.

Configurar el uso de los compiladores:

Para instalar un cierto compilador hay que utilizar el comando **module**. Por ejemplo:

```
$ module add gcc/10.2.0       // activa la versión 10.2 del compilador gcc
$ module add intel/2021.3.0   // activa la versión 2021 del compilador icc
$ module add nvidia-hpc-sdk/  // activa el compilador nvc de NVIDIA
```