Práctica 3
Paralelismo a nivel de hilos

Objetivos:

- Aprender a paralelizar una aplicación en una máquina paralela de memoria centralizada mediante hilos (*threads*) usando el estilo de variables compartidas.
- Estudiar el API de OpenMP y aplicar distintas estrategias de paralelismo en su aplicación.
- Aplicar métodos y técnicas propios de esta asignatura para estimar las ganancias máximas y la eficiencia del proceso de paralelización.

Entrenamiento previo:

Observe el siguiente programa en C donde se suman dos vectores de *floats* empleando OpenMP para paralelizar el cálculo.

```
#include <omp.h>
#define N 1000
#define CHUNKSIZE 100

main(int argc, char *argv[]) {

        int i, chunk;
        float a[N], b[N], c[N];

        /* Inicializamos los vectores */
        for (i=0; i < N; i++)
          a[i] = b[i] = i * 1.0;
        chunk = CHUNKSIZE;

        #pragma omp parallel shared(a,b,c,chunk) private(i)
          {
                  #pragma omp for schedule(dynamic,chunk) nowait
                  for (i=0; i < N; i++)
                    c[i] = a[i] + b[i];

          }   /* end of parallel region */

}
```

Revise la documentación de OpenMP (API, tutoriales, este enlace: https://computing.llnl.gov/tutorials/openMP/, etc...) y responda:

- ¿Para qué sirve la variable `chunk`?
- Explique **completamente** el *pragma* "`#pragma omp parallel shared(a,b,c,chunk) private(i)`". ¿Por qué y para qué se usa `shared(a,b,c,chunk)` en este programa? ¿Por qué la variable `i` está etiquetada como `private` en el pragma?
- ¿Para qué sirve `schedule`? ¿Qué otras posibilidades tiene?

Desarrollo:

En esta práctica los/las estudiantes deben paralelizar usando OpenMP una cierta aplicación para aprovechar los distintos núcleos de los que dispone cada ordenador de prácticas. Se paralelizará por tanto para un sistema multiprocesador (máquina paralela de memoria centralizada), en el que todos los núcleos de un mismo encapsulado ven la misma memoria, es decir, un puntero en un núcleo es el mismo puntero para el resto de núcleos del microprocesador.

Tarea 1: Estudio del API OpenMP **[Parte individual obligatoria (25% de la nota)]**

Se deberá estudiar el API de OpenMP y su uso con GNU GCC, comprobando el correcto funcionamiento de algunos de los ejemplos que hay disponibles en Internet (p.e. ejemplo https://lsi.ugr.es/jmantas/ppr/ayuda/omp_ayuda.php)

Realice un pequeño tutorial a modo de "libro de recetas de paralelismo con OpenMP" con distintos ejemplos de aplicación del paralelismo que ofrece OpenMP en función de distintas estructuras de programa.

Tarea 2: Propuesta en firme de una aplicación a paralelizar: **presentación + debate**

Cada grupo dispone de **3 minutos** para presentar el problema propuesto en la práctica anterior. Se deberá hacer hincapié en porqué es un buen candidato para su paralelización en OpenMP.
Cuando todos los grupos hayan expuesto, se realizará una votación para elegir el mejor candidato. Finalmente, **se debatirá** sobre la idoneidad del problema propuesto.

Tarea 3: Paralelización de la aplicación en OpenMP

Todos los grupos de cada turno de prácticas deben acometer la paralelización del problema elegido. Para ello se utilizará OpenMP para transformar el algoritmo y que incorpore paralelismo a nivel de hilos. El grupo que consiga mejorar más la ganancia recibirá puntuación extra.

Finalmente, se harán pruebas que comparen los resultados del algoritmo sin paralelizar y el algoritmo con paralelismo a nivel de hilo. Es importante que se justifique **lo más detalladamente** posible los cambios que se realicen para paralelizar el algoritmo. El **análisis de rendimiento tendrá que ser detallado** y lo más exhaustivo posible (probar varias cargas computacionales, ...).

Además, la memoria deberá dar clara respuesta a las siguientes cuestiones:

- Caracterización de la máquina paralela en la que se ejecuta el programa. (pe. Número de nodos de cómputo, sistema de caché, tipo de memoria, etc.).
  En Linux use la orden: `cat /proc/cpuinfo` para acceder a esta información.
- ¿Qué significa la palabra `ht` en la salida de la orden anterior? ¿Aparece en su ordenador de prácticas?
- Calcular lo siguiente (con gráficas asociadas):
  - Ganancia en velocidad (*speed-up*) en función del número de unidades de cómputo (*threads* en este caso) y en función de los parámetros que estime oportunos en su

aplicación y que modifiquen el **tamaño del problema** (p.e. dimensiones de una matriz, máximo error permitido, tamaño de algún kernel de convolución, etc...).
Comente los resultados de forma razonada. ¿Cuál es la ganancia en velocidad máxima teórica?
Nota: Puede entregar gráficas en 2D o 3D según resulte más ilustrativo.

- Calcular y representar de igual forma la eficiencia en función de los mismos parámetros que el caso anterior.
  Responda de forma justificada: ¿Cuál es la implementación más eficiente?

- **Es obligatorio el uso de distintos tipos de paralelismo** (de datos, funcional, …).
- Realice una representación gráfica de las tareas en forma de grafo de control de flujo que ilustre cómo es su proceso de paralelización.
- **Revise la documentación de la sesión 7 de teoría** *Paralelismo: conceptos y técnicas.* Aclare, para la paralelización que propone en su práctica lo siguiente:
  - Tipos de paralelismo usado.
  - Modo de programación paralela.
  - Qué alternativas de comunicación (explícitas o implícitas emplea su programa).
  - Estilo de la programación paralela empleado.
  - Tipo de estructura paralela del programa.

Notas generales a la práctica:

- No se deberá implementar ninguna paralelización a nivel de proceso.
- La implementación realizada tendrá que poder ejecutarse bajo el sistema operativo Linux del laboratorio de prácticas, aunque en casa puede trabajar con otros compiladores y sistemas operativos que soporten OpenMP (icc, clang, Visual C++, ...)
- Las/los estudiantes entregarán, además de la aplicación desarrollada, una memoria, estructurada según indicaciones del profesor, con la información obtenida.
- La información referente a OpenMP se encuentra en www.openmp.org. Para compilar use gcc o g++ con el modificador `-fopenmp`.
- **Es obligatorio** entregar un *Makefile* con las reglas oportunas para compilar y limpiar su programa de rápida y sencilla.
- La práctica se deberá entregar mediante el método que escoja su profesor de prácticas antes de la sesión de prácticas de la semana del 12 de noviembre.

Nota:

Los trabajos teórico/prácticos realizados han de ser originales. La detección de copia o plagio supondrá la calificación de "0" en la prueba correspondiente. Se informará la dirección de Departamento y de la EPS sobre esta incidencia. La reiteración en la conducta en esta u otra asignatura conllevará la notificación al vicerrectorado correspondiente de las faltas cometidas para que estudien el caso y sancionen según la legislación (Reglamento de disciplina académica de los Centros oficiales de Enseñanza Superior y de Enseñanza Técnica dependientes del Ministerio de Educación Nacional BOE 12/10/1954).