# Práctica 2
# Propuesta y estudio de paralelización

**Objetivos**:

- Revisar problemas y aplicaciones derivadas de **cualquier rama del conocimiento**, actuales o no, cuya **carga computacional sea tan elevada** como para justificar acometer un proceso de paralelización. Elegir una aplicación **secuencial** candidata a ser paralelizada.
- Defender mediante métricas ya conocidas en otras asignaturas porqué la aplicación en cuestión es idónea para su paralelización.
- Proponer arquitecturas idóneas para la paralelización de su aplicación.
- Estudiar cómo pueden afectar los parámetros de compilación al rendimiento de la aplicación en una máquina paralela.
- Aplicar métodos y técnicas propios de esta asignatura para estimar las ganancias máximas y la eficiencia del proceso de paralelización.

**Desarrollo**:

- **Tarea 1:** En esta primera práctica los/las estudiantes deberán afrontar un proceso de **búsqueda y selección de un problema**, de **cualquier índole y rama del conocimiento** (ingeniería, matemáticas, físicas, química, psicología, etc.), cuya solución venga dada en función de una aplicación software de elevado coste computacional. El problema en cuestión puede estar formulado desde cualquier ciencia o rama del conocimiento en general, y la búsqueda de información y consultas se deberá hacer por los cauces habituales: revistas, libros, Internet, tutorías, etc.
  **Nota**: El/la estudiante no tiene porqué entender cómo se ha resuelto el problema formalmente, pero sí que debe entender la estructura de su implementación y su dinámica.
  El programa debe estar escrito en C o C++ secuencial (es decir, no debe estar paralelizado) y **no debe ser interactivo**, es decir, tomará unos parámetros al inicio de su ejecución (línea de órdenes o un archivo de texto) y al cabo de un tiempo entregará los resultados. En caso de optar por un programa interactivo (por ejemplo el juego del ajedrez), se deberán enfrentar dos instancias del mismo programa para hacer el conjunto no interactivo y poder aplicar las métricas habituales.
  Nota: Esta tarea debe finalizarse en 2-3 horas de tiempo de aula de prácticas, al final de las cuales deberá comunicar el problema a su profesor para que le dé el visto bueno al problema y poder seguir con el resto de tareas.

- **Tarea 2: Estudiar la codificación del programa**, describiendo con máximo detalle cómo se ha resuelto el problema.
  - Esta codificación puede ser propuesta por el/la estudiante o tomada de algún recurso bibliográfico debidamente referenciado.
  - Analizar el programa mediante un **grafo de control de flujo (CFG)** (ver: https://www.youtube.com/watch?v=9N5vPeSWRfQ). ¿Revela el CFG "trozos" de aplicación cuya ejecución pueda ser acometida de forma paralela?
  - ¿Cómo es el acceso y cómo son las variables que usa la aplicación (grandes matrices de datos, datos que raramente se acceden...)? ¿Se pueden **estructurar** de forma más

apropiada para una ejecución más eficiente en una máquina paralela? ¿Podemos prever algún problema con la caché?
  - ¿Permite la aplicación **variar la carga** del problema para poder hacer un estudio más exhaustivo del rendimiento de su implementación?
- **Tarea 3**: La implementación realizada tendrá que ejecutarse bajo el sistema operativo Linux (Ubuntu 16.04 de la EPS) y el compilador GCC.
  - Estudie y refleje en una tabla los parámetros de compilación que piense que más puedan beneficiar al rendimiento de la aplicación.
    **TIP**: Revise qué hace "`gcc -help=target`", y "`gcc --help=optimizers`" en https://gcc.gnu.org/onlinedocs/gcc/Overall-Options.html

- **Tarea 4**: Análisis del rendimiento de la aplicación.
  - ¿Cómo se afecta el *speed-up* (ganancia en velocidad) si variamos el parámetro X de nuestro problema?
  - Genere gráficas del tipo: *Speed-up* o eficiencia *versus* parámetro_que_escala_nuestro_problema.

- **Entregables**:

  - Memoria estructurada de la práctica con **TODOS** los apartados siguientes debidamente trabajados (siga las recomendaciones de su profesor de prácticas).
    **Índice obligatorio**: objetivos, presentación del problema propuesto, estudio pormenorizado de la aplicación propuesta, análisis de rendimiento, defensa del programa ¿por qué es un buen candidato a paralelizar?, Arquitecturas paralelas idóneas para la ejecución del programa debidamente paralelizado, bibliografía.
  - Fichero comprimido en ZIP con un ***Makefile*** y los fuentes necesarios para la compilación del programa (si no es absolutamente necesario no entregue nunca binarios).
    Implemente por lo menos las reglas siguientes en su fichero m*akefile*:
    `make clean` (limpiar directorio de trabajo)
    `make` (construir el proyecto)
    `make run` (ejecutar el programa con los parámetros por defecto).

- La práctica se deberá entregar mediante el método que escoja su profesor de prácticas **antes** de la sesión de prácticas de la semana del 15 de octubre. No obstante, la práctica terminará después de la primera hora de la clase de práctica de la semana del 8 de octubre (**duración 5 horas**).

**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).*