FUNDAMENTOS DE SISTEMAS PARALELOS

Práctica 1: Programa MPI simple y caracterización del rendimiento

Objetivo

- Familiarizarse con la programación con MPI en problemas computacionales altamente paralelos.
- Utilizar mecanismos para validar su rendimiento, escalabilidad y precisión del resultado.
- Diseñar un informe riguroso sobre los resultados obtenidos.

Enunciado

- Esta práctica se realizará por parejas de alumnos, de manera que se abordarán soluciones diferentes para obtener con la mayor pecisión posible el valor del número π a través de métodos eminentemente paralelizables: el método de montecarlo para calcular áreas, o el cálculo de áreas por trapecios de funciones cuya integral esté relacionada con π, o el cálculo de algún sumatorio o productorio.
- Valor de referencia para π:
  3.14159265358979323846264338327950288419716939937510582097494 46

  Usa doble precisión para almacenarlo.
- Los métodos considerados para el cálculo de π se basan en las siguientes ecuaciones:

$$\pi = \left(\int_{-\infty}^{\infty} e^{-x^2} dx \right)^2 \tag{1}$$

$$\pi = 4 \int_0^1 \sqrt{1 - x^2} dx \tag{2}$$

$$\pi = \int_{-1}^{1} \frac{dx}{\sqrt{1 - x^2}} \tag{3}$$

$$\pi = 2 \prod_{k=1}^{\infty} \frac{(2k)^2}{(2k-1)(2k+1)} \tag{4}$$

$$\pi = \sqrt{\sum_{k=1}^{\infty} \frac{6}{k^2}} \tag{5}$$

$$\pi = \sum_{k=0}^{\infty} \frac{1}{16^k} \left(\frac{4}{8k+1} - \frac{2}{8k+4} - \frac{1}{8k+5} - \frac{1}{8k+6} \right) \tag{6}$$

$$\pi = \frac{22}{7} - \int_0^1 \frac{x^4 (1-x)^4}{(1+x)^2} dx \tag{7}$$

$$\pi = \frac{99^2}{\sqrt{8} \sum_{k=0}^{\infty} \frac{(4k)!(1103+26390k)}{(k!)^4 396^{4k}}} \tag{8}$$

$$\pi = 4 \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{2k-1} \tag{9}$$

$$\pi = \frac{1}{3} \int_{-3}^{3} \frac{x+3}{\sqrt{9-x^2}} dx \tag{10}$$

- Los métodos a considerar para calcular integrales son:
  - A: Método de Montecarlo: url
  - B: Método de los trapecios: url
- Cada grupo programará los métodos indicados por el profesor, y analizará su tiempo de ejecución T (en segundos, y en doble precisión) y el error E cometido en el cálculo de π (diferencia, en doble precisión, entre el valor de referencia y el obtenido). La calidad del resultado vendrá definida como: $C = \frac{1}{T \cdot E}$. Las medidas de tiempo de ejecución deben incluir todo el código salvo las operaciones de entrada/salida. Puedes usar la función de medida de tiempos *MPI_ WTime()* con *MPI_ Wtick()* o bien *gettimeofday()*. Ten en cuenta el *overhead* de las funciones de medida de tiempos.
- Se medirán los resultados de T, E y C para diferente número de procesadores, desde 1 hasta 32. Y se calcularán las mejoras obtenidas en T, E y C respecto al caso de un solo procesador.
- Usa una estrategia SPMD (Single Program Multiple Data) para elaborar el programa. Es decir, todos los procesos ejecutarán el mismo programa sobre diferentes datos. Usa el *rank* de cada proceso para determinar el trabajo específico que debe hacer. Si se necesitan números aleatorios, asegúrate de que cada proceso genera una secuencia diferente.
- Para la comunicación entre procesos solamente se podrán utilizar las funciones de envío y recepción de mensajes punto a punto que considerdes que se ajustan mejor a la solución propuesta. Recuerda que su coste es alto y conviene realizar el menor número de comunicaciones posible.
- Analiza la escalabilidad con el tamaño problema definido como el número de iteraciones del lazo principal.
- El entregable constará de varios ficheros separados (sin comprimir):
  - Un pdf por cada método en el que se comente brevemente el código y se muestren y analicen los resultados.
  - Los códigos fuente en ficheros .c de texto plano.
- De manera **optativa** puedes elegir otra forma de calcular π diferente