Práctica

2

Multiprocesadores

Utilización de la librería de paso de mensajes MPI.

En esta práctica se trabajará con las comunicaciones punto a punto asíncronas (no bloqueantes).

Ejercicio 1. Resolución de bloqueos

En el apartado 1.D de la práctica 1 viste que en uno de los casos el programa se bloqueaba. Debes cambiar las llamadas bloqueantes por no bloqueantes y resolverlo. Este ejercicio sencillo te permitirá usar las llamadas no bloqueantes y ver las diferencias con las bloqueantes.

Ejercicio 2. Rendimiento de las llamadas asíncronas

Aprovechando que tenemos el programa de la práctica 1 (1.D) de una pareja de procesos comunicándose, cámbialo para mandar un número de enteros parametrizado `T`. Un proceso será emisor y otro proceso receptor. De este programa realizaras dos versiones: una síncrona y otra asíncrona (no bloqueante).

En ambas versiones, el proceso emisor realizará cálculos, enviará esos cálculos al segundo proceso, volverá a realizar cálculos y volverá a enviarlos. Para simplificar el programa, los cálculos se simularán con una llamada al sistema `sleep()`, la primera fase será una espera de 1 segundo y la segunda de 6 segundos. El proceso receptor seguirá el mismo esquema: también hará una serie de cálculos, recibirá el mensaje, volverá a calcular y recibirá el otro mensaje. En este caso los tiempos son inversos al anterior, se esperará primero 6 segundos y después 1 segundo.

En la segunda versión cambiarás las llamadas de comunicación síncronas(bloqueantes) por llamadas asíncronas (no bloqueantes). Ten en cuenta que si no te interesa el resultado de cómo ha resultado el envío puedes usar la constante MPI_STATUS_IGNORE.

Teniendo en cuenta lo que hemos visto del tamaño del mensaje, deberás hacer las siguientes pruebas:

1. Piensa lo que se tardaría en ejecutar la primera versión con un número T pequeño y la versión síncrona. Si el tamaño T aumenta y supera el umbral, piensa cuánto tardará.

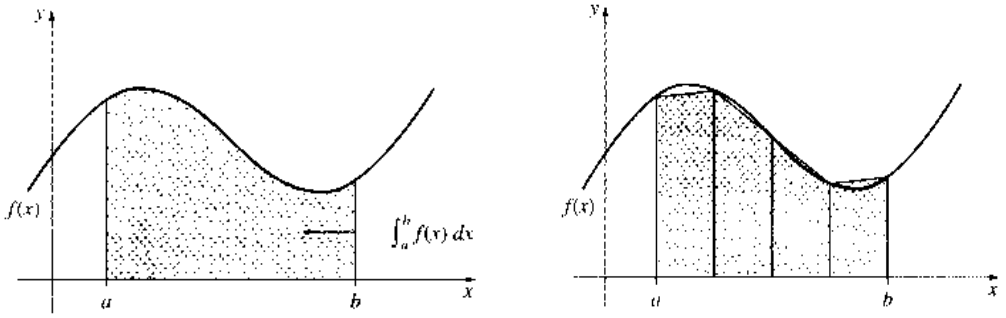
2. Ejecuta el programa síncrono, observa los resultado e indica si has acertado o no. En cualquier caso, explica los mismos.

3. Haz lo mismo que en 1 pero en el caso asíncrono.

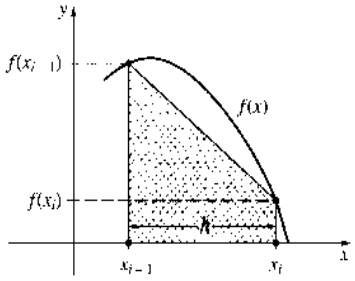
4. Haz lo mismo que en 2 pero en el caso asíncrono.

Ejercicio 3. Comparando paradigmas y plataformas

Ya que sabemos instrumentalizar un programa, en este ejercicio se trata de comparar los diferentes paradigmas de realización de un programa paralelo comparado con el secuencial, se trata de calcular el valor de una integral definida (cálculo del área) usando el método de los trapezoides.



La idea general es que dividimos la curva de la función en múltiples trapezoides de anchura h y una altura media (suma trapezoidal de Riemann) con el valor de la función en el valor inicial y final de x. El área total será la suma de las áreas de los trapezoides. Cuanto más estrecha sea h (más trapezoides haya) más exacta será la medida. Pero debes tener en cuenta que, en cualquier programa de cálculo numérico, se producen propagaciones de errores y que, pasado un punto, los resultados serán peores y se gastará más CPU tontamente.



En este caso no tienes que programar nada, sólo medir resultados. Se te dará el programa secuencial (Moodle) al que deberás incorporar una función más o menos compleja, ya que se te da la más simple que es la de la rampa. Y poner unos límites de la función y una anchura determinada de acuerdo a la potencia de tu computador (se trata de que el programa tarde lo suficiente para poder ser medido).

Por ejemplo, para $\int_0^{\pi} \cos x e^{\operatorname{sen} x}\,dx$ te tiene que dar 0.

También se te da una versión realizada con hilos POSIX sobre la que deberás aplicar las mismas modificaciones.

Por último, se te da una versión MPI que deberás modificar igualmente y que deberás ejecutar con de varias maneras, sacando una gráfica de speed-up variando las CPU.

Debes pensar en la versión de MPI cómo se hace la entrada de datos a, b y h. Si en todos los procesadores con un *scanf*, en uno único y transmitirlo o en forma de constantes.

No te sorprendas de los resultados, ya que estamos usando una versión de MPI sobre memoria compartida con pocos cores. Para poder sacar rendimiento a un programa paralelo con MPI, este debe ser muy requirente de cómputo de tal manera que se compense la introducción de la comunicación y la librería de MPI con la adicción de cores en diferentes máquinas.