P2.- Ejercicios de MPI

UNIVERSIDAD
DE GRANADA

Nicolás Calvo Cruz
Dpto de Arquitectura y Tecnología de los Computadores
@ncalvocruz
ncalvocruz@ugr.es

Índice

Bases

- Las prácticas se realizan en Linux nativo
- No se hacen en Windows
- No se hacen en una Máquina Virtual con Linux
- Se puede usar cualquier máquina, ATCGrid o personal (*)
- Las prácticas se entregan en swad.ugr.es en la fecha indicada.
- Entregar tarde reduce la nota (pero es mejor que no entregar)

Ejercicio 1 (1 punto)

- Una empresa tiene un potencial de N clientes. Para llevar sus cuentas guarda en un vector de tamaño N el número de ingresos realizados por sus clientes, mientras que en otro guarda el número de gastos de estos. Considera que los datos almacenados en los vectores son números aleatorios de tipo entero. Realiza un programa utilizando MPI que calcule la suma de todos los ingresos, la suma de todos los gastos y reste las cantidades. Para el diseño ten en cuenta que el procesador 0 es el que conoce los datos. Él será el encargado de distribuir la carga y de mostrar el resultado final.

Ejercicio 2 (1 punto)

- Haz un programa que busque el mínimo de un vector de valores reales (double) aleatorios en paralelo. La longitud del vector se debe indicar por parámetro. La carga de trabajo debe balancearse de la forma más equilibrada posible. El procesador 0 es el que tiene toda la información sobre el vector. El procesador 1 será el encargado de recoger la información final y mostrarla por pantalla. Impide que el programa se pueda ejecutar con menos de dos procesos.

Ejercicio 3 <I> (1 punto)

- Haz un programa que sólo considere la creación de 4 procesos MPI, cada uno con una funcionalidad distinta. El proceso 0 se encargará de la interacción con el usuario. Éste solicitará, en bucle, un comando que consistirá en un número (en el rango [0,4]). Dependiendo del número indicado en el comando, se realizará una de estas actividades:
  - Si es “0”, salimos del programa. Si es un número distinto de “0”, se ejecutará la acción correspondiente, se mostrará el resultado y se solicitará de nuevo otro comando.

Ejercicio 3 <II>

- (...):
  - Si es “1”, el proceso 0 solicitará por teclado la introducción de un texto y lo enviará al proceso 1, que pondrá el texto en mayúscula (función toupper). Se lo devolverá al 0, que lo mostrará por pantalla.

  - Si es “2”, el proceso 0 le enviará al 2 un vector con 10 números reales {1.1, 2.2,3.3….10.10}. Éste calculará la suma de todos ellos y la raíz cuadrada del resultado. Finalmente, se lo devolverá al 0, que imprimirá el resultado por pantalla.

Ejercicio 3 <III>

- (...):
  - Si es “3”, el número se enviará al proceso con identificador 3. Este escribirá por pantalla el mensaje “*Entrando en funcionalidad 3*”, calculará el entero correspondiente a cada letra del mensaje escrito, hará una suma de todos ellos y enviará el resultado al proceso 0, el cual lo mostrará por pantalla.

  - Si es “4”, el mensaje se enviará a todos los procesos, que harán la acción correspondiente a su identificador.

Ejercicio 4 (1 punto)

- Haz un programa que calcule la suma total de los elementos (reales) de una matriz NxM aleatoria. El tamaño de la matriz se indicará por consola. El proceso 0 será el encargado de crear la matriz, distribuir la carga y mostrar el resultado final. El reparto ha de hacerse de forma equilibrada y buscando la máxima velocidad.

Ejercicio 5 (3 puntos*)

- Realiza con MPI un programa que calcule el producto de 2 matrices de valores reales. El proceso 0 recibe como parámetro el tamaño de las matrices (filas de A, columnas de A y columnas de B), las inicializa, y finalmente se realiza el cálculo entre todos los procesos disponibles debiendo quedar el resultado final en el proceso 0. Este mostrará siempre el tiempo real que ha tardado la operación. Además, si las matrices son pequeñas (p.ej., hasta 7x7), se mostrarán A, B y el resultado del producto.

Ejercicio 6 <I> (3 puntos*)

- El laboratorio científico *Alchemy* ha desarrollado una sustancia química que, esparcida siguiendo un patrón determinado sobre la superficie de los metales, puede llegar a convertirlos en oro. Sin embargo, la sustancia es muy cara y potencialmente peligrosa si se usa de forma incorrecta. Por suerte, la plataforma de inyección es muy precisa y registra por zonas la dosis de sustancia liberada en extensos vectores de microgramos. El equipo del laboratorio está trabajando sobre un modelo de distribución adecuado para las superficies en las que se aplica.

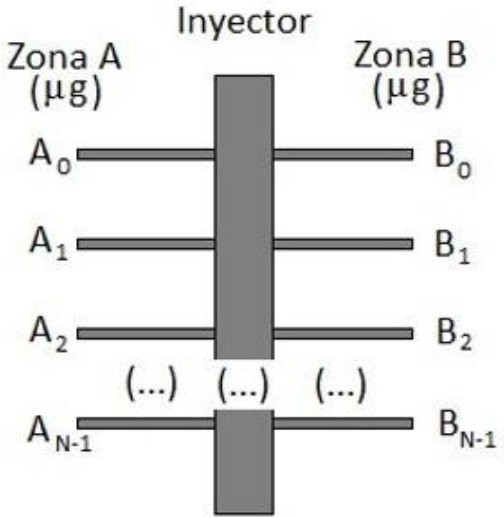
Ejercicio 6 <II>

- Para el controlador se necesita calcular, en un cierto instante de tiempo t, la similitud coseno [1] de los vectores de dosis aplicadas con respecto al eje de esparcimiento del tubo de inyección para ajustar la liberación de sustancia del instante siguiente t+1.

[1] https://en.wikipedia.org/wiki/Cosine_similarity

Ejercicio 6 <III>

- La siguiente figura da una idea aproximada de la situación descrita:



$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|}$$

Ejercicio 6 <IV>

- Se te ha contratado para desarrollar el módulo software que, aprovechando la plataforma de cálculo paralelo de memoria distribuida del laboratorio, pueda calcular la similitud coseno de los vectores de distribución en el menor tiempo posible. Debes preparar entonces un programa que calcule la similitud coseno de vectores de valores reales rápidamente.

- El jefe del proyecto informa de que el prototipo debe trabajar con valores reales (de tipo “double”) y que el rango operativo del proceso químico se mueve entre los 0 y 40 microgramos.

Ejercicio 6 <V>

- Como no se tiene acceso a datos reales, los vectores se escribirán aleatoriamente por el proceso 0. El programa recibirá por consola la longitud de los vectores. Finalmente, considerando que los usuarios finales no están acostumbrados a trabajar con programas en consola, es recomendable que se filtren mínimamente los parámetros recibidos para, por ejemplo, evitar longitudes negativas.

- Por consola se mostrarán tanto el valor devuelto como el tiempo de ejecución, en segundos. Igualmente, si los vectores son de tamaño menor o igual a 5, ambos se visualizarán junto con el resultado.

Ejercicio 7 (*)

- Escoge el ejercicio 5 ó el 6, y estudia el rendimiento que obtienes con respecto a la versión secuencial del programa. Para hacerlo:
  - Varía el tamaño de la entrada hasta que el programa secuencial tarde, al menos, entre 15 y 20 segundos. Registra las pruebas que hagas en este sentido.
  - Identificado un tamaño adecuado, estudia el rendimiento del programa paralelo cuando lanzas 2, 4, 8 y 16 procesos en concurrencia real (sírvete de ATCGrid si es necesario) para ese tamaño. Esto implica hacer una gráfica de aceleración y otra de eficiencia.
  - No olvides promediar cada medición de tiempo con, al menos, 5 tomas de tiempo.
  - Comenta los resultados. ¿Merece la pena la paralelización?

Entregables

- Se espera **recibir todos los códigos** desarrollados en .c, junto con un **documento que explique el enfoque de paralelización** seguido para los ejercicios 1 a 5.

- Para el ejercicio 6, en el mismo documento, se incluirán las **mediciones, gráficas y comentarios sobre el rendimiento** que se observa.

Fecha de entrega

- Esta práctica debe entregarse mediante SWAD el 20 de abril a las 15:30.

- Se abrirá una tarea a tal efecto.

¡Vamos a trabajar!