Seminario práctico 2

Introducción a la Interfaz de Paso de Mensajes (MPI)

Jose Luis Gallego Peña

Ejercicio 1.

Para la realización de este ejercicio se ha optado por crear dos comunicadores nuevos: uno para identificadores de proceso pares y otro para los impares. Entonces, los envíos y recepciones sólamente se hacen dentro del comunicador que le corresponde.

Para crear los dos comunicadores nuevos se usa *MPI_Comm_split*, siendo el color que se usa para dividir el resto de la división del rango global del proceso entre 2, lo cual nos indica si entra dentro de los pares (color 0) o impares (color 1). Para asegurarse que el proceso global 0 sea el 0 en los pares, y que el global 1 sea el 0 en los impares, indicamos como clave el propio rango global, para que tenga prioridad al obtener el nuevo identificador.

El resto es igual que en el ejemplo de send/receive sólo que usando el nuevo comunicador y mandando el rango que reciben al siguiente (nuevo rango más 1).

Ejercicio 2.

En este ejercicio se asignan subintervalos a cada proceso. Todos los procesos reciben el número total de subintervalos, y según su id de proceso calculan, según el valor de subintervalo, desde donde empiezan y donde acaban en el bucle para intentar que todos los procesos iteren el mismo número de veces. Se comprueba el caso de que el último proceso pueda tener un número menor de subintervalo al resto, o directamente 0, comprobando si la iteración del bucle que les corresponde es mayor al total de subintervalos que se tienen, y restándole el sobrante para que hagan menos iteraciones (para que hagan las iteraciones que quedan).

Por ejemplo, en el caso de tener 3 procesos y 4 iteraciones, cada proceso calcularía 2 iteraciones:

- El proceso 0 calcula la iteración 1 y 2.
- El proceso 1 calcula la iteración 3 y 4.
- El proceso 2 calcularía la iteración 5 y 6, sin embargo, 6 es mayor que 4 (el número total de iteraciones), por tanto especifica que calcula hasta la iteración 4 (la última), pero como tiene registrado que empieza en la 5, pues no llega a ejecutar el bucle directamente y su valor de PI es 0.

Se muestra por mensajes en pantalla desde donde hasta donde hace la iteración cada proceso.

Finalmente el proceso 0, tras realizar el *MPI_Reduce* con el que obtiene el valor completo de PI, realiza un *MPI_Bcast* para mandarles ese mismo valor a todos los procesos y que también lo tengan y muestren por pantalla.

Ejercicio 3.

Para este ejercicio simplemente se ha eliminado el VectorB local y se ha trabajado con un VectorB más pequeño, de tamaño acorde a la parte que le corresponde a cada proceso. Cada proceso inicializa ese vector más pequeño y lo usa para operar, ya que el proceso 0 solo inicializa el vector A y por tanto se elimina el *MPI_Scatter* del vector B y sólo queda el del A, que sí se reparte. Finalmente se hace el reduce con los datos obtenidos.

Ejercicio 4.

Para este ejercicio se han creado dos nuevos vectores. Un vector en el que están todos los valores, cuyo tamaño es el número de procesos impares, y para asegurarse de esto sólo el proceso con rango 1 global se ocupa de redimensionarlo y rellenarlo. El segundo vector es un vector local que tienen todos los procesos globales, de tamaño uno y con valor rellenado a 0.

Para realizar el MPI_Scatter y que sólo lo ejecuten los procesos impares, se filtra por id de proceso global y sólo entran los que tienen un id de proceso global impar, entonces se hace el Scatter dentro de el comunicador comm, evitando así que los pares obtengan el valor. Cada proceso impar recibe un único dato del vector.

Finalmente muestran, además de todo lo del ejemplo, el valor que han recibido del vector. Si es 0, es que no han recibido nada por ser pares, si es cualquier otro número generado aleatoriamente es porque son impares y han recibido el valor.