Ejercicios MPI(II)

Ejercicio 1

En este ejercicio se nos pedía que distribuyamos la matriz A[12][12] desde P0 a todos los procesadores en bloques bidimensionales. Es decir, una parte fundamental del ejercicio es utilizar la función MPI_Type_create_subarray() para crear dichos bloques y después poder repartirlos. Entonces, en el punto del código 1. Definición de un bloque 2D, eso es lo que hemos hecho, hemos calculado la variables Kbl, número de columnas-filas por bloque, van a ser iguales ya que el bloque es cuadrado ya que vamos a trabajar con 4-9-16. Y Kdim, que indica el número de bloques por fila o columna que tenemos. Una vez creadas estas dos variables realizamos los cálculos necesarios para pasarle a la función MPI_Type_create_subarray() los valores necesarios.

En el punto del código 2. Repartir A por bloques lo que va a consistir es que el pid = 0 vaya haciendo los cálculos necesarios para repartir los bloques correspondientes a cada proceso. Decidí que como el pid = 0 ya tiene los datos el for() de envíos empezase por 1 y así ahorramos recursos y tráfico. La otra parte del punto es el MPI_Recv() en los demás procesos para recoger los bloques.

En el punto del código 3. Repartir A por bloques mediante un scatter realizamos un MPI_Type_Create_resized() ya que queremos repartir los bloques mediante un scatterv(), por lo que necesitamos cambiar el tipo y que coja las columnas contiguas, en este caso Kbl. Después hacemos el cálculo de cuántos bloques vamos a repartir a cada proceso, que va a ser 1 y las distancias. Por último printeamos la matriz que le ha llegado al proceso P0, esto lo he hecho porque si printeas todas las matrices, se van a pisar unas a otras y no se entiende nada, por lo que printeo la del 0 y si quiero printear la matriz de otro proceso no tengo más que cambiar la condición del if por el pid que quiero printear.

Ejercicio 2

En este ejercicio se trata de ejecutar una aplicación de 32 procesos los cuales tienen que ser repartidos en cuatro comunicadores de 8 procesos. Además, los procesos P0 de cada comunicador deben formar otro comunicador, es decir una parte fundamental de este ejercicio es saber crear y construir comunicadores para llegar a ejecutar la aplicación.

Lo primero que encontramos en el código es un control de errores ya que este programa solo acepta que lo ejecutemos con npr = 32. Después el P0 inicializará la matriz A.

En el punto 1. Generar los grupos lo que haremos será crear estos grupos a través del valor de la variable colour, el cual va a controlar el grupo al que van y key que va a marcar el orden en el cual están en el nuevo comunicador que definamos. Además de eso les he asignado un subcomunicator para que en los print() se vea más claro a qué grupo pertenecen. Por último, creo el nuevo comunicador llamado new_comm y el new_pid.

En el punto 2. Repartir la matriz a los “jefes” de grupo lo que hago es que el P0 del comunicador global reparta a los P0 de las cabecera de los nuevos comunicadores la matriz por filas, una a cada cabecera. Para ello, lo que hago es que mediante una condición de pid == 0 el reparto de P0 y otra condicional con new_pid == 0 para que las cabeceras recojan la información. Se que podría haber creado el comunicador de las cabeceras al principio y haber jugado con eso, pero no lo he visto necesario ya que solo lo necesito al final para hacer la recogida del mínimo de las sumas, por lo que lo he decidido así.

En el punto 3. Repartir los datos recibidos entre los miembros del grupo y procesarlos, lo que hacemos es un MPI_Scatter() dentro de los 4 comunicadores para que así el reparto se haga dentro de estos, y cada proceso dentro de los nuevos comunicadores recoja un trozo de la fila, en este caso, cada proceso recogerá 10 números. Luego, realizamos la operación rand() % (pid + 1), ya que no especifica qué hacer, y por último recogemos en las cabeceras otra vez la fila entera.

Por último, en el punto 4. Operación colectiva de suma entre los jefes de grupo, lo que hago es que las cabeceras de los grupos realicen la suma de las filas, y ahora sí creo el comunicador de las cabeceras para poder envíar el valor mínimo obtenido entre todas las sumas de las cabeceras a las mismas. Además las cabeceras imprimen dicha información, tanto la suma como el mínimo.