Ejercicios MPI(I)

1er ejercicio

Primera parte

En la primera parte había que repartir las tareas entre todos los procesos de modo estático consecutivo por lo que lo que hemos hecho es calcular cuántas tareas le corresponden a cada uno, de ahí la variable n_tareas, y luego en el for he tratado de que cada proceso empiece y termine donde debe para que se realicen todos los números.

Segunda parte

La segunda parte es muy importante entender el "protocolo" que establecemos mediante los tags. En el programa lo que he hecho es que el manager entre en un bucle el cual espera todo el rato a que le lleguen mensajes de parte de los workers hasta que ha enviado los 1000 números y hasta que los mensajes finales se han enviado.
Estos mensajes finales hacen que los workers le envíen los datos necesarios que piden en el ejercicio como el número que ha necesitado más interacciones y cuántas han sido. Cuántos números ha procesado cada worker ya lo controla el manager con el vector num_worker, y cada vez que le manda un numero a un worker le suma 1 a num_worker[pid]. El worker sin embargo consiste en enviar un mensaje de que está listo al manager y a partir de ahí entra en bucle para enviar mensajes y procesar números hasta que se agoten.

Los tags del manager son 2, el caso 0, el cual envía números a los workers hasta que ya haya enviado 1000, en tal caso, envía los mensajes finales. Por otro lado, el manager tiene el caso 1, es decir si le llega un mensaje con tag 1 significa que hay un worker que le ha enviado un mensaje de finalización con la información del número que más iteracciones ha tardado en procesar. Los workers también tienen dos casos, el caso 0, el cual es para que le envíen números, los cuales va a procesar y cuando acabe de procesarlo va a enviar un mensaje al manager para que le envie otro. Por otro lado está el caso 1, el cual significa que se ha acabado de procesar todos los números y que tiene que enviarle al manager los resultados obtenidos.

Tercera parte

	collatz-primera						**collatz-segunda**					
	2	**4**	**8**	**16**	**24**	**32**	**2**	**4**	**8**	**16**	**24**	**32**
Speed-Up	1.78	3.42	6.83	12.5	18.5	24.5	1	2.98	6.88	14.5	21.7	28.7
Eficiencia	0.89	0.85	0.85	0.78	0.77	0.76	0.5	0.74	0.86	0.9	0.95	0.89

2º Ejercicio

El ejercicio 2 es bastante secuencial dado que vamos necesitando el resultado del anterior para calcular el siguiente. En mi programa empiezo inicializando las variables Am y B como marca el enunciado. Posteriormente calculo las variables necesarias para hacer un scatterv de la matriz Am y que cada proceso tenga su cacho de la matriz y la calcule por separado al resto. Antes de hacer el scatterv hago el broadcast del vector B y luego realizo el scatterv de la matriz Am(no es por nada especial simplemente porque he elegido ese orden). Una vez que me aseguro que cada proceso ha recibido B y su cacho correspondiente de la matriz empiezo a calcular C sabiendo que C = Am * B.
Después de esto aprovecho para hacer el calculo de C^2 que voy a necesitar para más tarde y hago un reduce para que el P0 ya tenga el resultado de la sumatoria del vector C elevado al cuadrado. El siguiente paso que sigo es hacer un ghaterv de C para que P0 recoja todos esos cachitos que se han calculado en los diferentes procesos y posteriormente pueda hacer un broadcast de C a todos los procesos y asi poder calcular D, ya que sabemos que D = Am * C. Más tarde, hacemos otro reduce de D^2 ya que lo vamos a necesitar posteriormente. Por último P0 se encarga de sumar D^2 y C^2 y hacer la raíz para printearla. El Gatherv de D que hacemos al final no haría falta pero como en el programa normal nos dais los resultados de D[0], D[N/2] y D[N-1] pues lo hago para poder printear los resultados.