Informe Práctica III Paralelismo

Introducción

En esta práctica se parte de un código secuencial dado, el cual calculaba la similitud de dos secuencias genéticas. El objetivo de la práctica era realizar la paralelización del código mediante la descomposición del dominio en múltiples tareas y así repartir la carga de procesamiento entre los núcleos del procesador.

Realización de la práctica

Para realizar esta tarea, primero implementamos una versión simple dónde el número de filas de la matriz (N) fuese divisible entre el número de procesos (P), usando las funciones de *MPI_Scatter()* y *MPI_Gather()* para la distribución de las filas de la matriz entre los P procesos. En esta versión todos los procesos tenían la misma carga.

Para que el programa pudiese funcionar correctamente cuando N/P es distinto de 0, consideramos oportuno cambiar las funciones *MPI_Scatter()* y *MPI_Gather()* por *MPI_Scatterv()* y *MPI_Gatherv()*, su funcionalidad es la misma solo que en las últimas se puede distribuir la carga entre los procesos de manera más eficiente que con las otras funciones. Por ejemplo, con 7 procesos y 8 filas, con la solución inicial el reparto se haría cogiendo 2 filas por proceso, puesto que hay que redondear hacia arriba para no dejar filas sin coger, y los 4 primeros procesos habrían hecho todos los cálculos desperdiciando así 3 núcleos. Otra solución habría sido redondear la división hacia abajo y que las filas restantes las computase el proceso 0. En cambio, con *MPI_Scatterv()* y *MPI_Gatherv()* en el primer ejemplo el reparto de la carga sería de 2 filas para el proceso 0 y una fila para el resto, siendo así la mejor solución en cuanto a distribución de la carga.

Para el cálculo de tiempos de toman los tiempos antes y después de los envíos y durante la computación con la función *gettimeofday()*. Cada proceso guarda sus tiempos de comunicación y de computación, que son enviados al proceso 0, con la función *MPI_Gather()*, para que este los imprima por pantalla.

***Nota:** La ejecución del código se ha realizado en una máquina virtual Ubuntu 20.04 con 6 hilos y 8GB de RAM debido a problemas técnicos en la máquina en la que se realizó el desarrollo del código con Linux nativo. Debido eso los tiempos de ejecución son muy elevados.