Curso: Programación paralela y concurrente.
Estudiante: Fabiola Jiménez González, carnet B23452.

Respuestas al Problema 2. Jaccard_mpi del examen 2.

1. ¿Cuál o cuáles funciones de MPI consideró necesarias para la solución de este problema y explique cómo funciona(n) en su código?

Para mi solución utilicé las siguientes funciones de MPI:

MPI_Init, esta es la función que inicializa el ambiente de MPI en mi programa con los argumentos que recibe por parámetro en argc y argv.

MPI_Comm_size, esta función escribe sobre mi variable (num_processes) el valor del número de procesos que utilizará MPI para la ejecución paralela de mi programa. Esta función me sirvió para determinar en mi código la repartición de las líneas entre los procesos.

El proceso al cuál le tocaba analizar cada línea se calculó de acuerdo a la función:

número_de_iteración%(num_processes-1) (1)

MPI_Comm_rank, esta función escribe sobre mi variable (my_id) el valor del número de proceso que se está ejecutando.

Esta variable me sirvió en mi programa para sincronizar el proceso que debía analizar cada línea de la matriz. De acuerdo a la función (1), el proceso activo sería el cálculo de esta función. Si el id del proceso concordaba con este valor, entonces este se encargaba de analizar la línea de la iteración que se estaba ejecutando.

MPI_Allreduce, esta función se encarga de reducir los valores locales de max_jaccard_similarty de todos los hilos y colocar el máximo en la variable global largest_jaccard_found, de esta manera el programa encuentra cuál línea era más similar con la línea a evaluar. Además de que ayuda a notificarle al hilo que encontró el índice de jaccard más grande que tiene que imprimir, si el valor local de jaccard era igual al valor global máximo, entonces este hilo debe imprimir en consola.

MPI_Finalize, utilicé esta función para decirle a MPI que el ambiente de ejecución terminó.

2. Explique cómo realizó la repartición de tareas entre los procesos.

Utilicé una fórmula en la que se registra el número de iteración de lectura de la matriz, y a partir del número de iteración se determina el proceso activo realizando la operación (1), nuevamente:

Proceso_activo = número_de_iteración%(num_processes-1) (1)

De esta manera al entrar el en while de lectura, si mi_id es igual al número de proceso que debería estar activo, entonces realizo el for que cuenta la unión y la intersección.

De esta manera los procesos se van turnando la lectura de las líneas de la matriz. En pseudocódigo:

```
Inicializo el iterador;
While() //mientras hayan lineas por leer
      Si mi id es el proceso que debería estar activo{
            Compara y calcula la interseccion entre la linea que se
            esta analizando y la linea a evaluar
            Calcula el coeficiente local maximo de jaccard
      }si no{
            No hago nada
      }
      Incremento el iterador;
      Determino el siguiente proceso activo con la formula 1.
}
```