PRACTICA 1
Sistemas Concurrentes y Paralelos

José Miguel Avellana López 18068091G
Jassine El Kihal X8592934L

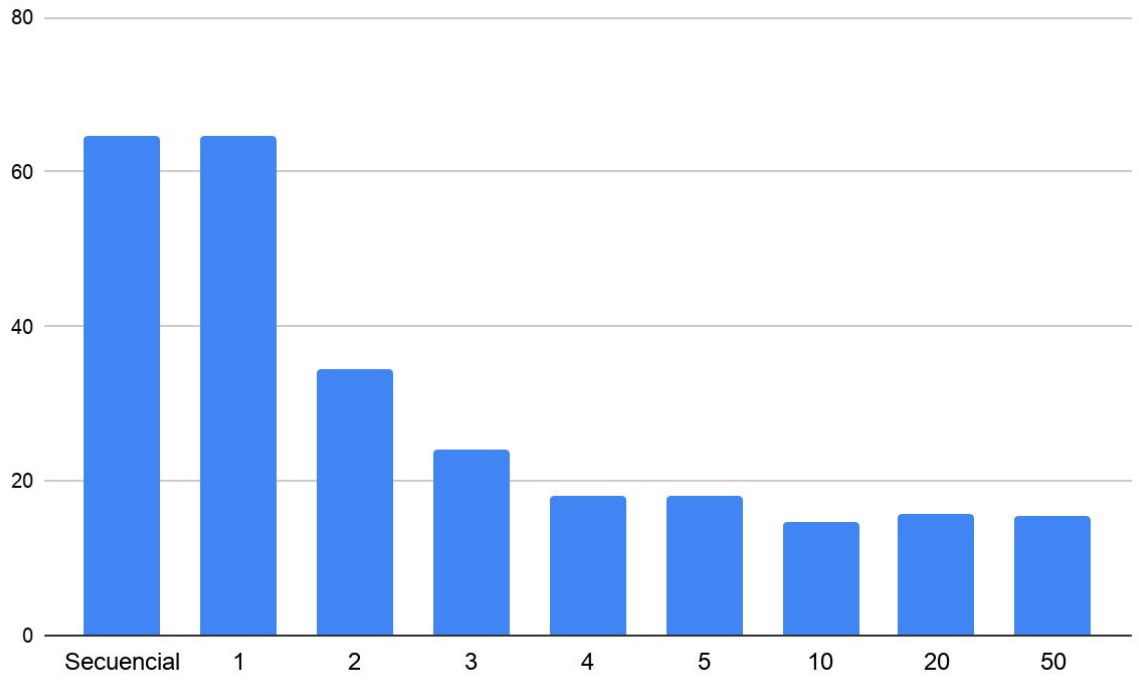
El objetivo de la practica es adaptar el código secuencial de un problema a una versión concurrente, dando uso de la libreria POSIX de C/Unix. De manera que el tiempo de ejecución de la version modificada sea menor que la original.

Para la resolucion del problema se generan y evaluan cada una de las posibles combinaciones con el fin de obtener la que posea los atributos mas optimos.

Con el fin de medir el tiempo de ejecucion del programa en sus distintas versiones y numero de hilos se da uso del comando Time de Unix.

De los tres tiempos que da el comando, *real* da el tiempo real de ejecucion sin tener en cuenta que puede haber otros programas ejecutandose en el sistema, *user* da el tiempo de ejecucion del programa (sin kernel) y sin contar procesos externos, y finalmente *sys* da el tiempo de las llamadas al kernel del proceso.

El problema es que *user* cuenta el tiempo de cada hilo de manera independiente y despues los suma, por lo tanto en un programa secuencial lo mejor seria user+sys, pero teniendo encuenta que queremos comprobar la eficacia de los hilos *se dará uso del tiempo real*.



El programa se ha testeado en un equipo con las siguientes caracteristicas: un i7-7700HQ de 4 nucleos y 8 hilos y una frecuencia básica de 2.8GHz.

Se puede observar que no hay demasiada diferencia entre la version secuencial y la concurrente con un solo hilo de ejecucion. A partir de alli, al duplicar el numero de hilos se decrementa el tiempo de ejecucion a la mitad, hasta llegar a cuatro hilos de ejecucion, punto en el que se observa un tiempo de ejecucion aparentemente constante.
Esto se debe a que el procesador puede, como máximo, ejecutar 4 hilos de manera simultánea, si aumentas el número de hilos a partir de ese punto el sistema se encargará de ejecutarlos de manera concurrente.

En cuanto a la forma en la que se dividen las combinaciones entre el numero de hilos es obteniendo la cantidad de combinaciones por bloque, siendo un bloque el numero de combianciones entre el numero de hilos mas uno.
El motivo por el que se suma uno al bloque es para que no exista riesgo de que puedan quedar combinaciones sin ningun bloque en el caso en que la division no sea entera, solo existe riesgo de que un bloque tenga menor número de combinaciones. Para ello se esa la variable *bloqueS,* se comprueba si la ultima combinacion se pasa del rango de combinaciones, y, de ser asi, se hace que la ultima combinacion del bloque sea la ultima del rango general.

Aunque este algoritmo garantiza que se cubran todo el rango de combinaciones, no garantiza que la reparticion de combinaciones entre los bloques este balanceada. A pesar de esto la diferencia con un gran número de combinaciones es inapreciable.