Guillem Escriba Molto - u188331

Sistemas Operativos

Informe Práctica 3 y 4

El objetivo esencial de ambas prácticas era el uso de threads para realizar operaciones de convolución de imágenes sin y con sincronización. Para ello se usan distintas funciones relacionadas con los threads, locks, en este caso barriers, y bucles para iterar.

En la práctica 3, en la segunda sección, la pregunta del apartado 7, tras probar con 1, 10, 20 y 30, aparte de aumentar el tiempo de ejecución sucesivamente, algo bastante lógico pues cada vez se realizan más iteraciones, se veía cada vez más distorsionado y con más pérdida de información. Por ejemplo, en los casos extremos, 1 y 30, en el primero, se veían los números difuminados, pero ya no se veían pixelados, en cambio, en el 30, los números ya ni se veían.

Respecto al benchmark de las 3 convoluciones, va variando bastante aún usando el mismo número de iteraciones, según tengo entendido, esto se debe a replit. A nivel teórico, el que menos tiempo debería tomar sería el tercero, ya que combina threads con sincronización lo que le debería permitir ser el más rápido, en cambio, el primero debería ser el más lento ya que un solo thread principal es el encargado de realizar todos los cálculos. Pero no es así, cada vez que se ejecuta los valores varían considerablemente. El segundo debería ser más rápido que el primero ya que al tener más threads, cuando a uno se le acaban el número de instrucciones que puede ejecutar en la CPU, en vez de tener que esperarse hasta que le vuelva a tocar, como pasaría en el primero, pasa otro thread y sigue realizando cálculos, y así sucesivamente. Por otro lado, el tercero, además de lo ya mencionado, no tiene que crear nuevos threads en cada iteración por qué todos los threads participan en todas las iteraciones.

A parte del tiempo que se ahorra al no tener que crearlos de nuevo, también permite que mientras que un thread va cambiando la imagen, el resto siga realizando operaciones.

En estas prácticas he tenido bastantes problemas, no en la primera, sino en la segunda, es decir, la práctica número 4. Si bien conseguí hacer el código más rápido que la primera, por alguna razón que desconozco, al resetear las barreras, se producía un deadlock o se interrumpía la ejecución del programa. He probado a poner prints por todas partes y parecía que se quedaba en pthread_cond_wait, pero no entiendo el porqué, ya que está como lo hicimos en la teoría. También he probado a cambiar la posición de las barreras y de los resets, pero están donde deben estar, ya que se necesita que bloqueen las líneas de código correspondientes. Por probar he probado hasta crear una barrera por iteración con su respectivo reset, pero tampoco funciona. Sin embargo, si no pongo los resets, si que "funciona" el problema es que las barreras así no son útiles ya que a partir de la primera iteración, cuando missing llegue a 0, en las siguientes iteraciones, como missing no volverá a ser n_threads, todo el rato, cuando un thread termine de hacer la convolución cambiará la imagen sobreescribiendo la de otros threads y por tanto no se realizará correctamente la convolución, al menos no en toda la imagen.

Exceptuando esto, el resto funciona adecuadamente, con cualquier número de threads, al menos potencias de 2, aunque también me ha dado resultados con 3 y 5 pero no he probado mucho más. La convolución se realiza de manera correcta en el primero y el segundo, y en el tercero no se corresponden el número de iteraciones a la convolución, pero es lo que digo, desconozco que es lo que sucede, ya que a mi parecer está todo correcto.