Sistemas Operativos
Laboratorio 3
Angel Higueros - 20460

Laboratorio #3

Responda las siguientes preguntas:

1. ¿Qué es una race condition y por qué hay que evitarlas?

   Es cuando dos o más procesos intentan acceder y modificar una variable compartida al mismo tiempo, lo que resulta en datos impredecibles o incorrectos. Se deben evitar porque pueden causar errores y posibles inconsistencias en el programa.

2. ¿Cuál es la relación, en Linux, entre pthreads y clone()? ¿Hay diferencia al crear threads con uno o con otro? ¿Qué es más recomendable?

   Son dos formas de crear threads en Linux. pthreads es una librería que proporciona una API para crear y manejar threads, mientras que clone() es una llamada al sistema para crear un proceso clonado. Sus diferencias están en la forma en la que se crean y manejan los threads. Dependiendo de cada caso se recomienda una u otra opción.

3. ¿Dónde, en su programa, hay paralelización de tareas, y dónde de datos?

   Hay paralelización de datos en los ciclos for de ambas funciones.

4. Al agregar los #pragmas a los ciclos for, ¿cuántos LWP’s hay abiertos antes de terminar el main() y cuántos durante la revisión de columnas? ¿Cuántos user threads deben haber abiertos en cada caso, entonces?

   Al agregar los #pragmas a los ciclos for, habría un número máximo de LWP's abiertos igual al número de threads solicitado con la función omp_set_num_threads(). Durante la revisión de columnas, habría 9 user threads abiertos. En el caso de main() limitado a un thread, habría 1 LWP abierto durante la revisión de columnas y antes de la limitación habría 9.

OpenMP crea por defecto un número de threads igual al número de procesadores disponibles.

5. Al limitar el número de threads en main() a uno, ¿cuántos LWP's hay abiertos durante la revisión de columnas? Compare esto con el número de LWP's abiertos antes de limitar el número de threads en main(). ¿Cuántos threads (en general) crea OpenMP por defecto?

   Cuando se limita a un solo thread en main(), habría 1 LWP abierto durante la revisión de columnas. OpenMP crea un número de threads igual al número de procesadores disponibles por defecto.

6. Observe cuáles LWP's están abiertos durante la revisión de columnas según ps. ¿Qué significa la primera columna de resultados de este comando? ¿Cuál es el LWP que está inactivo y por qué está inactivo? Hint: consulte las páginas del manual sobre ps.

   La primera columna de ps muestra los identificadores de procesos o PID. El LWP inactivo es el proceso principal de OpenMP.

7. Compare los resultados de ps en la pregunta anterior con los que son desplegados por la función de revisión de columnas per se.
   a. ¿Qué es un thread team en OpenMP y cuál es el master thread en este caso?

      Un thread team es un conjunto de threads que ejecutan una tarea determinada.

   b. ¿Por qué parece haber un thread "corriendo", pero que no está haciendo nada?

      Porque se encuentra en un busy-wait

   c. ¿Qué significa el término busy-wait?

Es un ciclo continuo de verificación de una condición sin hacer ninguna tarea útil.

d. ¿Cómo maneja OpenMP su thread pool?

   OpenMP maneja su threadpool asignando tareas a los threads según la política de planificación y el número de threads disponibles.

8. Luego de agregar por primera vez la cláusula schedule(dynamic) y ejecutar su programa repetidas veces, ¿cuál es el máximo número de threads trabajando según la función de revisión de columnas? Al comparar este número con la cantidad de LWP's que se creaban antes de agregar schedule(), ¿qué deduce sobre la distribución de trabajo que OpenMP hace por defecto?

   Luego de agregar la cláusula schedule(dynamic) y ejecutar el programa repetidas veces, el número máximo de threads trabajando durante la revisión de columnas es 9. Esto sugiere que la distribución de trabajo por defecto en OpenMP es bastante equilibrada.

9. Luego de agregar las llamadas omp_set_num_threads() a cada función donde se usa OpenMP y probar su programa, antes de agregar omp_set_nested(true), ¿hay más o menos concurrencia en su programa? ¿Es esto sinónimo de un mejor desempeño? Explique.
   hay menos concurrencia, ya que se limita el número de threads. Depende de cada caso, hay que tomar en cuenta la cantidad de recursos disponibles y lo que se desea resolver con el programa.

10. ¿Cuál es el efecto de agregar omp_set_nested(true)? Explique.

    Permite que los threads secundarios creen su propio conjunto de threads, lo que puede aumentar la concurrencia y mejorar el rendimiento