UVG
UNIVERSIDAD DEL VALLE DE GUATEMALA

Laboratorio III
Sistemas Operativos

Maria Jose Castro Lemus #181202

---

1. ¿Qué es una race condition y por qué hay que evitarlas?
    a. Suceden cuando un sistema trata de hacer dos o más operaciones al mismo tiempo, estas se deben evitar, ya que debido a la naturaleza del dispositivo las operaciones deben realizarse bajo una secuencia para que estas sean ejecutadas de forma efectiva. Si esto no sucede esto puede afectar el funcionamiento del sistema.

2. ¿Cuál es la relación, en Linux,entre pthreads clone()? ¿Hay diferencia al crear threads con uno o con otro?¿Qué es más recomendable?
    a. La función clone () genera o crea un nuevo proceso, es muy similar a como funciona el fork. Pero este crea un nuevo contexto para su ejecución, al punto de ser considerado un nuevo proceso o thread, depende el uso que se desee. Por otro lado los pthreads crean un nuevo pthread dentro del proceso donde este fue llamado.

3. ¿Dónde,en su programa,hay paralelización de tareas, y dónde de datos?
    a. El paralelismo de tareas se da con la creación de distintos hilos para cada una de las comprobaciones (filas, columnas, subgrids) porque son muchas tareas distintas ejecutándose sobre el mismo conjunto de datos (el sudoku). En cambio con el #pragama omp parallel for hay paralelismo de datos ya que un hilo se hace responsable de una columna pero cada hilo ejecuta las mismas operaciones sobre cada una de las distintas columnas.

4. Al agregar los #pragmas a los ciclos for, ¿cuántos LWP'shay abiertos antes de terminar el main()y cuántos durante la revisión de columnas?¿Cuántos user threads deben haber abiertos en cada caso, entonces? Hint:recuerde el modelo de multithreading que usan Linux y Windows.
    a. Hubo la misma cantidad de LWP's abiertos, la cual equivale a 3330.

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?
    a. Habían 12 y luego solo hace 1. OpenMP elige el número ideal de threads por default en base a la cantidad de cores, estos se distribuyen automáticamente y si la división entre cores no es exacta un core hará más o menos trabajo.

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.
    a. La primera columna indica el process flag. El primero tiene estatus 's' lo que dice que está Interruptible sleep (waiting for an event to complete) lo cual significa que está esperando a los demás threads que se completen y terminen de correr.

7. Compare los resultados de psen la pregunta anterior con los que son desplegados por la función de revisión de columnas per se.¿Qué es un thread teamen OpenMP y cuál es el master thread en este caso?¿Por qué parece haber un thread "corriendo", pero que no está haciendo nada? ¿Qué significa el término busy-wait?¿Cómo maneja OpenMP su thread pool?
    a. OpenMP crea una una colección de thead teams, en este caso el thread master de cada equipo es el encargado de ejecutar la región o segmento asignado.
    b. En el caso del thread que no hace nada tiene una serie de procesos que forman parte del team y hasta finalizar de ejecutarse todos procederá a ejecutarse el master, este crea un fork para que el resto de procesos del equipo realicen sus tareas.
    c. El término busy-wait nos dice que un thread está en espera a que se cumpla una condición para poder ejecutar su tarea.
    d. OpenMP maneja su thread pool con un complemento lleno de threads, este sirve para tener listos los threads que se usan durante el proceso. En OpenMP se establece la cantidad que es adecuada para que el proceso que se desee verificar.

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?

a. El número de threads fue 5. Esto se puede deducir que al tener libertad de ejecución de número de threads puede definirse la eficiencia de ejecución

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.
    a. Hubo menos concurrencia porque se puede ver que imprimió menos hilos generados, por lo cual considero que su desempeño fue menos eficiente. Depende de cuantos threads fueron asignados openMP busca usar la mayor cantidad de threads posibles de acuerdo al número de procesadores. Entonces si le asignas la misma cantidad de threads que usaba OpenMP al inicio el resultado debe ser el mismo.

10. ¿Cuál es el efecto de agregar omp_set_nested(true)? Explique.
    a. Según la documentación esta función funciona para habilitar relaciones paralelas anidadas. Esto permite crear una relación paralela a la ya existente, también permite ajustar el tamaño de los threads teams.