Universidad del Valle de Guatemala
Sistemas Operativos
Tomás Gálvez Peña

Andrea María Cordón Mayén
Carné: 16076
Fecha: 25/03/2019

Laboratorio #5
"Multithreading"

1. **¿Qué es una race condition y por qué hay que evitarlas?**
    a. Una race condition ocurre cuando dos o más threads pueden ingresar a datos compartidos e intentan cambiar los datos al mismo tiempo. Es importante evitarlas ya que si esto llegara a pasar se perdería la data que uno de los threads necesita y por lo mismo que un proceso necesita. También porque causaría problemas de calendarización y descontrolaría todo.

2. **¿Cuál es la relación entre pthreads y clone()? ¿Hay diferencia al crear threads con uno o con otros? ¿Qué es más recomendable?**
    a. Clone() es una llamada especial del sistema que se utiliza mayor mente para implementar threads (especialmente pthreads).
    b. La diferencia que existe entre crear threads con pthreads y con clone es que cuando se crear un nuevo thread utilizando pthreads es como que se estuviera creando un nuevo thread que comparte la misma memoria que todos los demás, mientras que si se utiliza clone es como que si se creara un nuevo proceso que tendrá su propio espacio de memoria.
    c. No es que utilizar uno se más recomendable que el otro, todo depende de qué es lo que se quiere hacer: Multithreading o multiprocessing.

3. **¿Dónde, en su programa, hay paralelización de tareas y dónde datos?**
    a. Cuando se utiliza OpenMP para paralelizar el programa es cuando se están paralelizando tareas. Posteriormente, cuando se crea un nuevo thread para analizar los datos del sudoku, entonces se paralelizan los datos.

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

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 crea OpenMP (en general) por defecto?**
    a. Después de limitar el número de threads a uno hay 7 LWP's. Antes de limitarlo, habían 4 LWP's. Al final hay 4 threads que corresponden a la cantidad de cores que se le asignó a la máquina virtual al inicio del laboratorio.

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 inactivo y por qué está inactivo?**

7. **Compare los resultados de ps en la pregunta anterior con los que son desplegados por el método de revisión de columnas per se. ¿Qué es un thread team en 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. Un thread team en OpenMP es el conjunto de threads que se asignarán para ejecutar un proceso o task en específico.
    b. Porque no todos los threads posibles se encuentran ejecutándose.
    c. Busy waiting significa: es una técnica en la que un proceso revisa constantemente si una condición se cumple o no. También puede utilzarse para agregar un delay de tiempo.

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 el método de revisión de columnas? Al comparar este número con la cantidad de LWP's que se creaban antes de agregar Schedule(), ¿qué sucede sobre la distribución de trabajo que OpenMP hace por defecto?**
    a. El máximo número de threads debería corresponder al máximo de cores que posea la máquina virtual, en este caso 4 threads.
9. **Luego de agregar las llamadas omp_set_num_threads() a cada método donde se usa OpenMP, y luego de ejecutar su programa con y sin cláusula Schedule() en cada for, ¿hay más o menos concurrencia en su programa? ¿Es esto un sinónimo de un mejor desempeño? Explique.**

10. **¿Cuál es el efecto de agregar omp_set_nested(true)? Explique.**
    a. Nested parallelism permite que el programador pueda crear una región paralela dentro de la región misma. Esto significa que cuando le mandamos true a esta función, hacemos que sea permitido el paralelismo dentro del programa y la ejecución del mismo. Es decir que permitimos que los grupos que se hayan creado puedan crear nuevos grupos entre sí.