**Laboratorio No 1 Introducción al paralelismo – hilos**

Santiago Cárdenas Amaya

Juan Pablo Fonseca

Escuela Colombiana de ingeniería Julio Garavito

Bogotá 2023

Introducción

En el día a día los computadores vienen con procesadores con la famosa tecnología multi-core, la cual permite que una mismo CPU pueda ejecutar más de una acción al tiempo, dando así un boost de rendimiento en procesos complejos o procesos multi- ples. Así mismo a nivel de codigo se puede simular una función similar para bajarle carga a un programa usando las famosos thread o hilos lo que permite dividir la car- ga de una funcionalidad. De esta forma se puede reducir el tiempo de ejecución en varios casos por minutos cosa que la hará mucho más eficiente, en cualquier caso. ¿o no?

# Desarrollo del laboratorio – parte I

1. De acuerdo con lo revisado en las lecturas, complete las clases CountThread, para que las mismas definan el ciclo de vida de un hilo que imprima por pantalla los números entre A y B.
2. Complete el método main de la clase CountMainThreads para que:
   1. Cree 3 hilos de tipo CountThread, asignándole al primero el intervalo [0..99], al segundo [99..199], y al tercero [200..299].
   2. Inicie los tres hilos con 'start()'.
   3. Ejecute y revise la salida por pantalla.

Imagen que contiene Texto

Descripción generada automáticamente

* 1. Cambie el inicio con 'start()' por 'run()'. ¿Cómo cambia la salida?, ¿por qué?

La salida cambió y ahora los métodos se ejecutan secuencialmente. Esto se debe a que el método start() es el que inicia un Thread, y como no fue así en este caso, el código tendrá un flujo secuencial

Imagen que contiene Texto

Descripción generada automáticamente

# Desarrollo del laboratorio – parte II

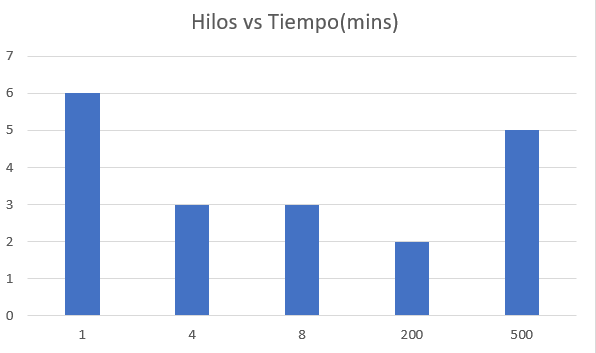
1. Cree una clase de tipo Thread que represente el ciclo de vida de un hilo que calcule una parte de los dígitos requeridos.
2. Haga que la función PiDigits.getDigits() reciba como parámetro adicional un valor N, correspondiente al número de hilos entre los que se va a paralelizar la solución. Haga que dicha función espere hasta que los N hilos terminen de resolver el problema para combinar las respuestas y entonces retornar el resultado. Para esto, revise el método join del API de concurrencia de Java.
3. Ajuste las pruebas de JUnit, considerando los casos de usar 1, 2 o 3 hilos (¡este último para considerar un número impar de hilos!).

# Desarrollo del laboratorio – parte III

A partir de lo anterior, implemente la siguiente secuencia de experimentos para calcular el millon de dígitos (hex) de PI, tomando los tiempos de ejecución de los mismos (asegúrese de hacerlos en la misma máquina):

1. Un solo hilo.
2. Tantos hilos como núcleos de procesamiento (haga que el programa determine esto haciendo uso del API Runtime).
3. Tantos hilos como el doble de núcleos de procesamiento.
4. 200 hilos.
5. 500 hilos.

Para realizar este punto se realizó una prueba en PiCalcTest.java, y a partir de los resultados obtenidos se elaboró la siguiente gráfica.



1. ¿Por qué el mejor desempeño no se logra con los 500 hilos?, cómo se compara este desempeño cuando se usan 200?

Esto se debe a que, aunque el proceso se divide más la salida no permite tantos ingresos a la vez provocando que sea más ineficiente después de cierto número de hilos seguir agregando, es por esto que con 200 se logra el mejor rendimiento y se demuestra con la ley de Amdahls.

1. ¿Cómo se comporta la solución usando tantos hilos de procesamiento como núcleos comparados con el resultado de usar el doble de éste?

La diferencia entre estas dos pruebas es de segundos, esto puede ser porque al usar todos los núcleos reales del pc se consigue el rendimiento más optimo posible

1. De acuerdo con lo anterior, si para este problema en lugar de 500 hilos en una sola CPU se pudiera usar 1 hilo en cada una de 500 máquinas hipotéticas, ¿la ley de Amdahls se aplicaría mejor? Si en lugar de esto se usaran c hilos en 500/c máquinas distribuidas (siendo c es el número de núcleos de dichas máquinas), se mejoraría? Explique su respuesta.

Si ya que todos los hilos actuarían al tiempo en vez de dejar la carga solo sobre un procesador y permitiría una mayor velocidad ya que los hilos acabarían al tiempo y la demora seria de la ilustración de los datos

Código del laboratorio: https://juanpablofonsecac5@dev.azure.com/juanpablofonsecac5/ARSW/\_git/ARSW

# Conclusiones

Se pudo concluir que el uso prudente y organizado de hilos puede traer mejoras significativas a un proyecto; a su vez, el mal uso de estos puede dificultar significativamente el rendimiento de la aplicación