Escuela Colombiana de Ingeniería

Arquitecturas de Software

Introducción al paralelismo - hilos

Kevin Jefrey Mendieta Perez.

Parte I Hilos Java.

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:
    i. Cree 3 hilos de tipo CountThread, asignándole al primero el intervalo [0..99], al segundo [99..199], y al tercero [200..299].
    ii. Inicie los tres hilos con 'start()'.
    iii. Ejecute y revise la salida por pantalla.
    iv. Cambie el inicio con 'start()' por 'run()'. Cómo cambia la salida?, por qué?. **Respuesta:** Al cambiar start() por run(), se deja de hacer paralelismo,es decir, se ejecutan secuencialmente cada unos de los llamados a los hilos (uno después del otro), como se hace secuencialmente, no hay paralelismo.

Parte II Hilos Java.

La fórmula BBP (Bailey–Borwein–Plouffe formula) es un algoritmo que permite calcular el enésimo dígito de PI en base 16, con la particularidad de no necesitar calcular nos n-1 dígitos anteriores. Esta característica permite convertir el problema de calcular un número masivo de dígitos de PI (en base 16) a uno vergonzosamente paralelo. En este repositorio encontrará la implementación, junto con un conjunto de pruebas.

Para este ejercicio se quiere calcular, en el menor tiempo posible, y en una sola máquina (aprovechando las características multi-core de la mismas) al menos el primer millón de dígitos de PI (en base 16). Para esto

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 o 2 hilos.

Parte III Evaluación de Desempeño.

A partir de lo anterior, implemente la siguiente secuencia de experimentos para calcular el millón 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.

Al iniciar el programa ejecute el monitor jVisualVM, y a medida que corran las pruebas, revise y anote el consumo de CPU y de memoria en cada caso. Con lo anterior, y con los tiempos de ejecución dados, haga una gráfica de tiempo de solución vs. número de hilos. Analice y plantee hipótesis con su compañero para las siguientes preguntas (puede tener en cuenta lo reportado por jVisualVM):

Número de procesadores de la CPU: 8

Número de hilos	**Tiempo de Solución(Segundos)**
1	1587s
8	420s
16	285s
200	246s
500	241s
1000	240s
1500	257s

1. Según la ley de Amdahls, donde *S(n)* es el mejoramiento teórico del desempeño, *P* la fracción paralelizable del algoritmo, y *n* el número de hilos, a mayor *n*, mayor debería ser dicha mejora. ¿Por qué el mejor desempeño no se logra con los 500 hilos?, cómo se compara este desempeño cuando se usan 200?.

   Respuesta: La fraccion paralelizable depende de la capacidad de procesamiento del equipo en el que se vaya a ejecutar el algoritmo, así como de otros factores, pero principalmente se puede deducir de la fórmula teórica que si esta fracción paralelizable no es muy grande, a partir de cierto número de hilos, el mejoramiento va incrementar muy poco (se puede evidenciar en las gráficas), y posteriormente empezara a ser negativo el incremento.

Como se dijo anteriormente incrementar la cantidad de hilos no implica que el mejoramiento en el tiempo sea mayor, y por esta misma razón no se logra un mejoramiento despues de cierta cantidad de hilos. En nuestro caso a partir de 200 hilos el desempeño empiesa a crecer muy poco, hasta el punto (1500 hilos) se empeora el tiempo de solución.

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

   Respuesta: Hay un mejoramiento en el tiempo de más de dos minutos. Por lo que la solucion usando tantos hilos como nucleos es mas lenta, ya que no necesariamente esta cantidad de hilos es la que da mejor desempeño, no obstante a partir de este punto se observa que el mejoramiento empieza a crecer muy lento.

3. 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.

   Respuesta: Teóricamente si existiría un mejoramiento, ya que serian 500 hilos en paralelo corriendo en diferentes computadores, por lo que la limitación de las maquinas no haría que el mejoramiento en el tiempo de solución empeore en vez de mejorar.

   Si el numero de nucleos de las máquinas es grande, entonces van a haber pocas máquinas con muchos nucleos, con lo que teóricamente (sin tener en cuenta los otros recursos de las máquinas) mejoraría el tiempo de solución. Si el numero de nucleos es pequeño, entonces habrían muchas maquinas con pocos núcleos, lo cual mejoraria el tiempo de solución.