

Andrés Felipe Ariza Pardo

Juan Pablo Sánchez Bermúdez

Introducción al paralelismo – hilos

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.

```
public class CountThread implements Runnable {  
    2 usages  
    private int A;  
    2 usages  
    private int B;  
  
    5 usages  Andrés Ariza  
    public CountThread(int A, int B) {  
        this.A = A;  
        this.B = B;  
    }  
  
    Andrés Ariza  
    @Override  
    public void run() {  
        for (int i = A; i <= B; i++) {  
            System.out.println(i);  
        }  
    }  
}
```

2. Complete el método main de la clase CountMainThreads para que:
 - a. Cree 3 hilos de tipo CountThread, asignándole al primero el intervalo [0..99], al segundo [99..199], y al tercero [200..299].
 - b. Inicie los tres hilos con 'start()'.

```

public class CountThreadsMain {

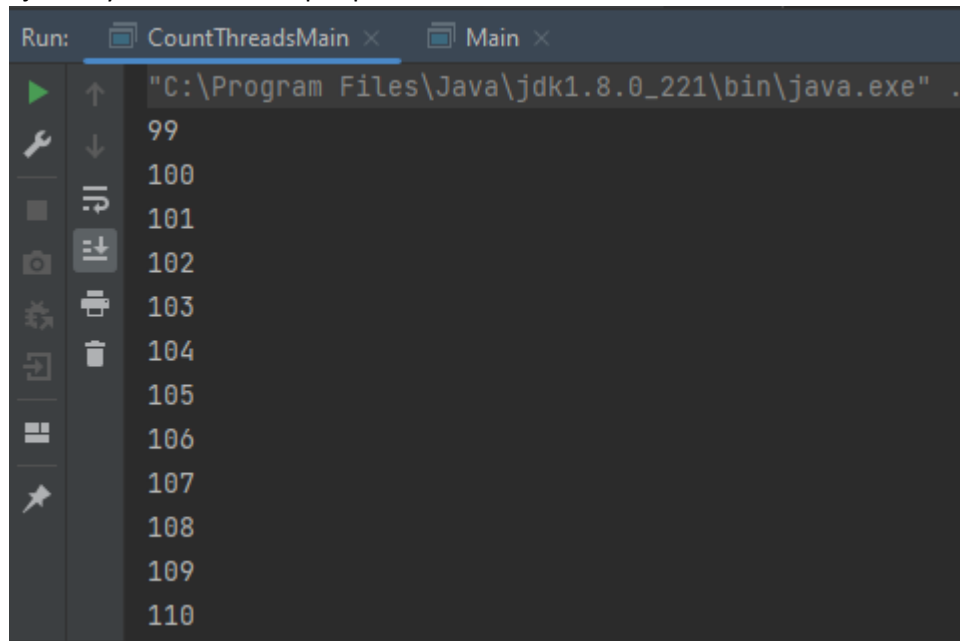
    Andrés Ariza +1

    public static void main(String a[]){
        Thread myThread0 = new Thread(new CountThread( A: 0, B: 99));
        Thread myThread1 = new Thread(new CountThread( A: 99, B: 199));
        Thread myThread2 = new Thread(new CountThread( A: 199, B: 299));
        myThread0.start();
        myThread1.start();
        myThread2.start();
    }

}

```

- c. Ejecute y revise la salida por pantalla.



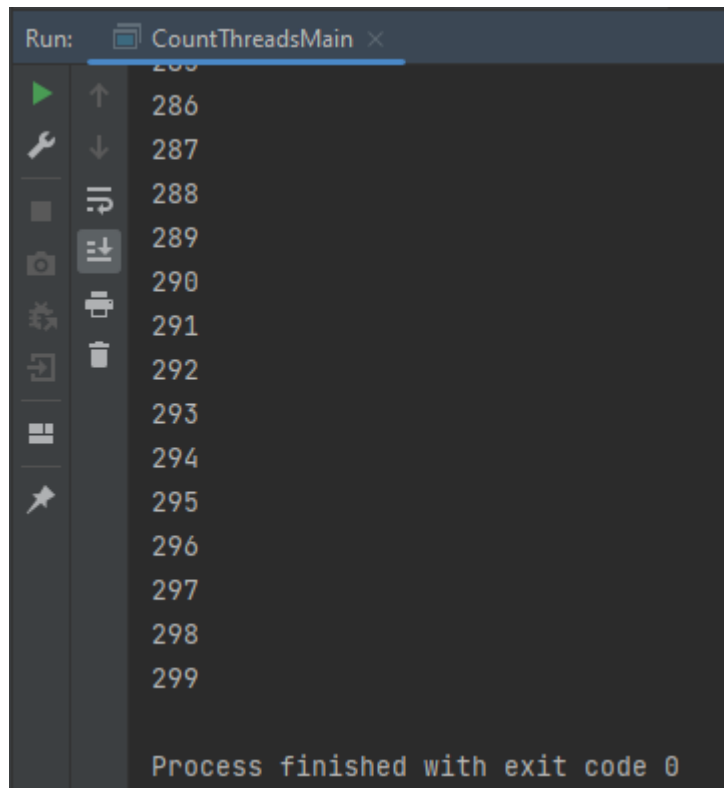
Run: CountThreadsMain x Main x

```

"C:\Program Files\Java\jdk1.8.0_221\bin\java.exe" .
99
100
101
102
103
104
105
106
107
108
109
110

```

- d. Cambie el inicio con 'start()' por 'run()'. Cómo cambia la salida?, por qué?



```
Run: CountThreadsMain x
286
287
288
289
290
291
292
293
294
295
296
297
298
299
Process finished with exit code 0
```

Se ejecutan secuencialmente.

Start crea un nuevo hilo y luego se llama al método run para ejecutar la acción en dicho hilo. Al usar run directamente, no se crea un nuevo hilo y la acción se ejecuta de la misma manera que se ejecutaría un método cualquiera, sin multithreading [1].

Parte II Hilos Java

1. Cree una clase de tipo Thread que represente el ciclo de vida de un hilo que calcule una parte de los dígitos requeridos.
Hicimos que la clase PiDigits implementara la interface Runnable, añadiendo dos atributos (start y count) para el rango de los dígitos de PI y el método bytesToHex (antes ubicado en Main) para poder visualizar el resultado en pantalla.
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.
Ya que hicimos que PiDigits fuera el hilo, creamos una clase “orquestadora” (PiDigitsMain), la cual crea los hilos y asigna los rangos para cada uno.
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!)

No realizamos pruebas Junit ya que el método run no retorna nada, así que realizamos pruebas para hallar los 50.000 primeros dígitos con 1, 2 y 3 hilos respectivamente. En la última línea de cada prueba se puede visualizar el tiempo (en ms) transcurrido en cada prueba.

```
"C:\Program Files\Java\jdk1.8.0_221\bin\java.exe" ...
243F6A8885A308D313198A2E03707344A4093822299F31D0082EFA98EC4E6C89452
60703

Process finished with exit code 0

"C:\Program Files\Java\jdk1.8.0_221\bin\java.exe" ...
243F6A8885A308D313198A2E03707344A4093822299F31D0082EFA98EC4E6C89452
49071

Process finished with exit code 0

"C:\Program Files\Java\jdk1.8.0_221\bin\java.exe" ...
243F6A8885A308D313198A2E03707344A4093822299F31D0082EFA98EC4E6C89452
37450

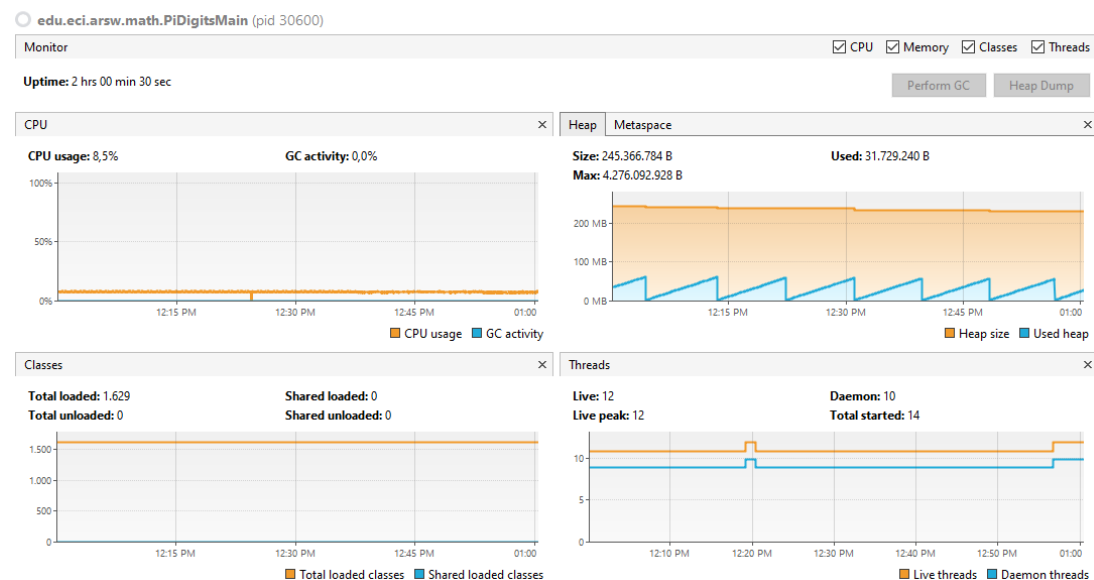
Process finished with exit code 0
```

Parte III Hilos Java

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):

Las pruebas se realizaron con los primeros 500.000 dígitos ya que 1 millón era demasiado.

1. Un solo hilo.



2. Tantos hilos como núcleos de procesamiento (haga que el programa determine esto haciendo uso del API Runtime).

```
System.out.println(Runtime.getRuntime().availableProcessors());
```

12

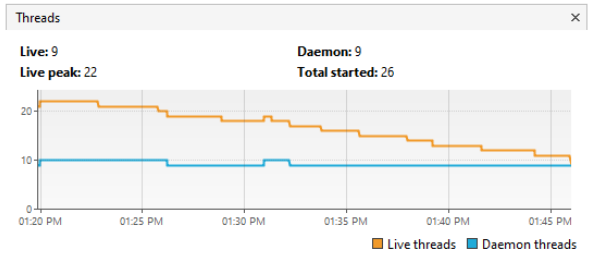
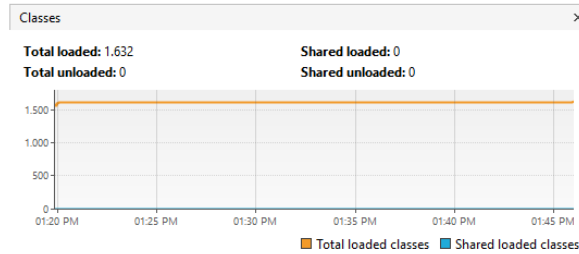
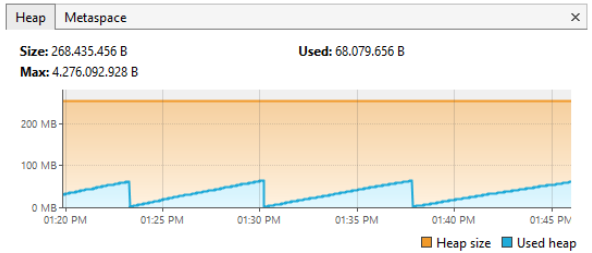
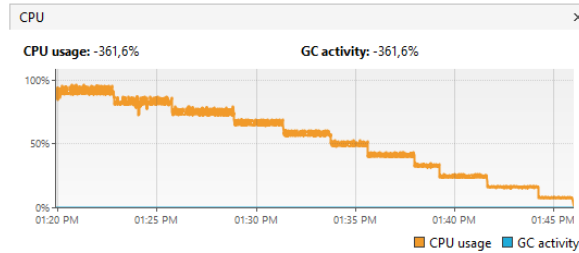
edu.eci.arsw.math.PiDigitsMain (pid 49820)

Monitor

☒ CPU ☒ Memory ☒ Classes ☒ Threads

Uptime: 27 min 28 sec

Perform GC Heap Dump



3. Tantos hilos como el doble de núcleos de procesamiento.

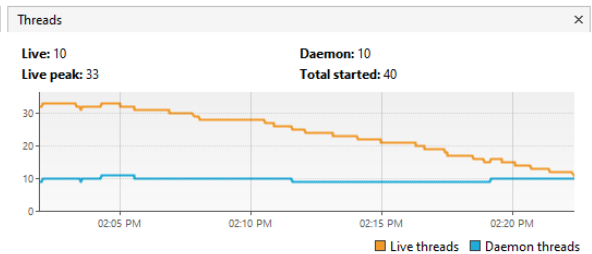
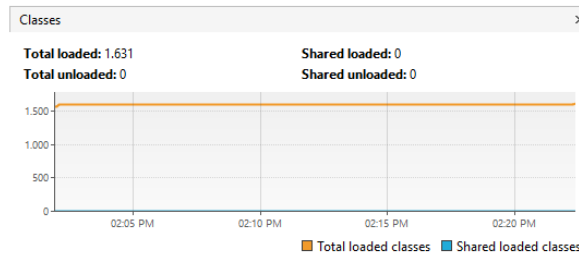
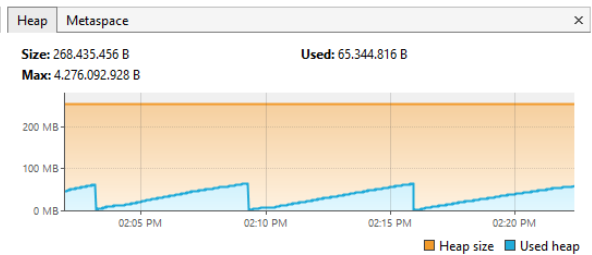
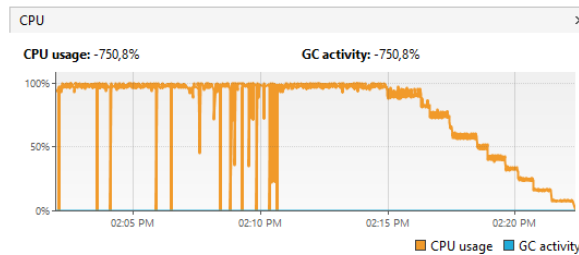
edu.eci.arsw.math.PiDigitsMain (pid 50508)

Monitor

☒ CPU ☒ Memory ☒ Classes ☒ Threads

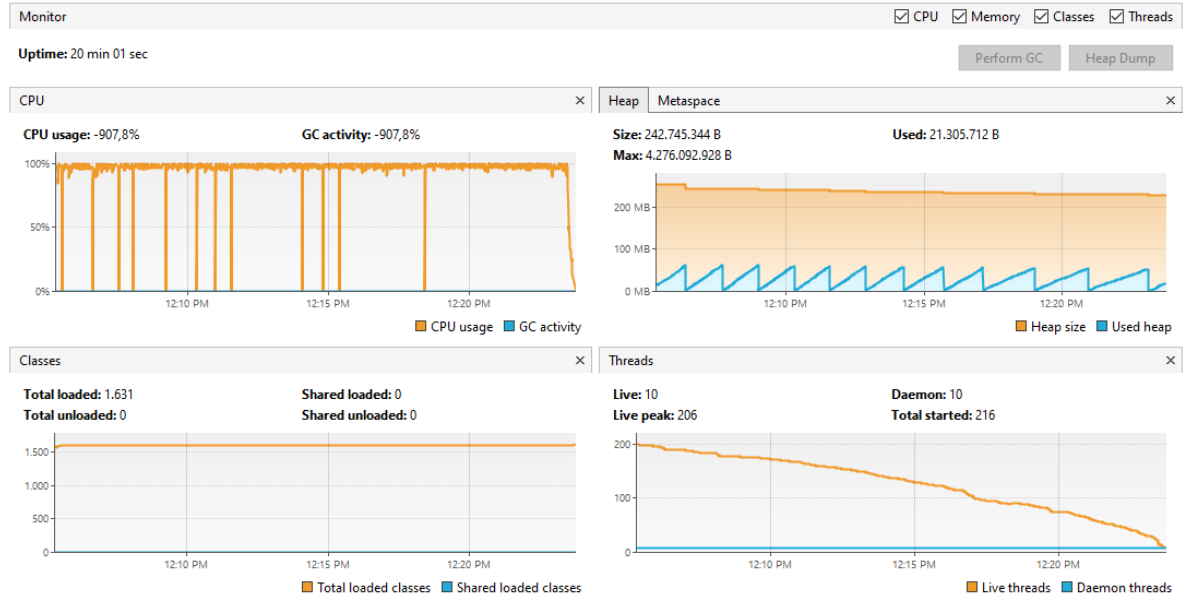
Uptime: 22 min 14 sec

Perform GC Heap Dump



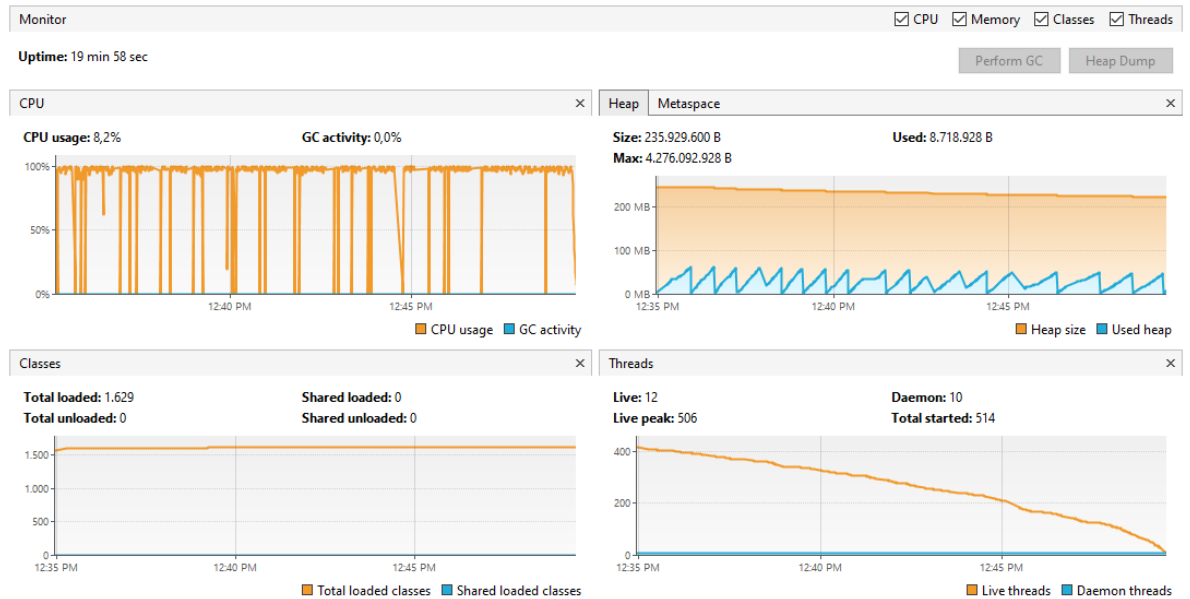
4. 200 hilos.

edu.eci.arsw.math.PiDigitsMain (pid 34400)

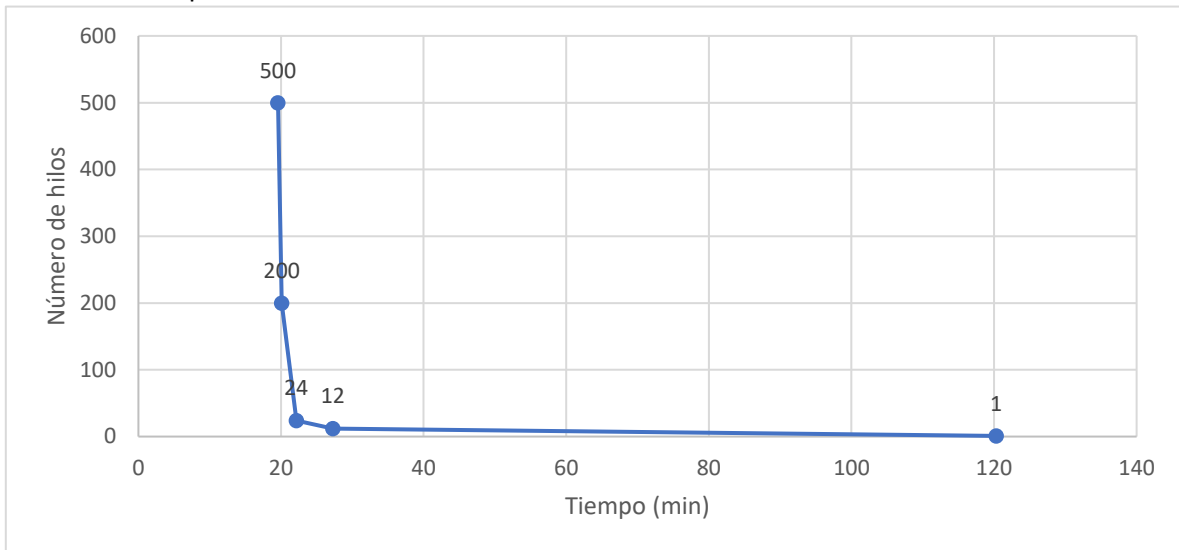


5. 500 hilos.

edu.eci.arsw.math.PiDigitsMain (pid 16564)



Gráfica de tiempo de solución vs número de hilos



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?

Según la ley de Amdahl no se obtendrá el mismo beneficio al ir añadiendo más y más hilos, por lo que el rendimiento usando menos hilos se ejecutará a una frecuencia más alta que usando más hilos a una frecuencia baja.

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

Con 12 hilos (cantidad de núcleos de procesamiento del procesador) se obtuvo un tiempo de 27 minutos con 28 segundos. Al usar el doble de estos se obtiene una mejora notable marcando un tiempo de 22 minutos con 14 segundos, pero se evidencia un consumo más elevado del uso de estos núcleos de procesamiento lógico.

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.

Referencias

- 1) <https://www.codingninjas.com/codestudio/library/difference-between-run-and-start-method>
- 2) <https://www.pugetsystems.com/labs/articles/Estimating-CPU-Performance-using-Amdahls-Law-619/#WhatIsAmdahlsLaw?>
- 3)