

Problema de programación paralela

La suma de los números primos menores a 10 es:

$$2 + 3 + 5 + 7 = 17$$

Utilizando el lenguaje de programación indicado por tu profesor (Scheme, Racket, Clojure), escribe dos versiones de un programa que calcule la suma de todos los números primos menores a 5,000,000 (cinco millones):

- La primera versión debe ser una implementación convencional que realice el computo de manera secuencial.
- La segunda versión debe realizar el computo de manera paralela a través de los mecanismos provistos por el lenguaje siendo utilizado (por ejemplo *p/aces* o la función *pmap*). Debes procurar paralelizar el código aprovechando todos los núcleos disponibles en tu sistema.

Ambas versiones del programa deben dar 838,596,693,108 como resultado.

Con el fin de que el proceso de cómputo sea más intenso para el CPU, utiliza el siguiente algoritmo:

Algoritmo para determinar si n es un número primo. Devuelve *verdadero* o *falso*.

1. Si n es menor que 2, el algoritmo termina devolviendo *falso*.
2. Para i desde 2 hasta $\lfloor \sqrt{n} \rfloor$, realiza lo siguiente:
 - El algoritmo termina devolviendo *falso* si n es divisible entre i de manera exacta, de otra se repite el ciclo con el siguiente valor de i .
3. El algoritmo termina devolviendo *verdadero* si el ciclo del punto anterior concluyó de manera normal.

Mide el tiempo en que tarda en ejecutar cada versión del programa y calcula el *speedup* obtenido usando la siguiente fórmula:

$$S_p = \frac{T_1}{T_p}$$

En donde:

- p es el número de procesadores (o núcleos).

- T_1 es el tiempo que tarda en ejecutarse la versión secuencial del programa.
- T_p es el tiempo que tarda en ejecutarse la versión paralela del programa utilizando p procesadores.
- S_p es el *speedup* obtenido usando p procesadores.

Escribe un breve documento en donde reportes los resultados obtenidos y entrégalo junto con el código fuente de tus implementaciones.

Realizamos 5 pruebas:

1. 1 Hilo - 3228 milisegundos
2. 10 Hilos - 708 milisegundos
 - a. $S_{10} = 3200 \text{ ms} / 708 \text{ ms} = 4.52$
3. 100 Hilos - 551 milisegundos
 - a. $S_{100} = 3200 \text{ ms} / 551 \text{ ms} = 5.80$
4. 1000 Hilos - 595 milisegundos
 - a. $S_{1000} = 3200 \text{ ms} / 595 \text{ ms} = 5.37$
5. 1500 Hilos - 681 milisegundos
 - a. $S_{1500} = 3200 \text{ ms} / 681 \text{ ms} = 4.69$
6. 10000 Hilos - 1009 milisegundos
 - a. $S_{10000} = 3200 \text{ ms} / 1009 \text{ ms} = 3.17$

Entre 20 y 100 hilos es lo más óptimo que encontramos, sin embargo, de 20 hasta mil hilos no existe gran diferencia, de 1500 hacia arriba ya no es óptimo.

Código de nuestro programa

MiHilo.java

```
public class MiHilo extends Thread {
    int id;
    int ciclos;
    boolean bandera;
    int firstNum;
    int lastNum;
    private long resultado;

    MiHilo(int id, int ciclos, boolean bandera, int firstNum, int lastNum) {
        this.id = id;
        this.ciclos = ciclos;
        this.bandera = bandera;
        this.firstNum = firstNum;
        this.lastNum = lastNum;
    }

    private boolean numberIsPrime(int num) {
        if(num < 2){
            return false;
        }
        else {
            boolean numIsPrime = true;
            for(int i=2; i <= Math.sqrt(num); i++) {
                if(num % i == 0) {
                    numIsPrime = false;
                    break;
                }
            }
            return numIsPrime;
        }
    }

    @Override
    public void run() {
        System.out.println("Hilo " + id + " empieza");
        for(int i=this.firstNum; i < this.lastNum; i++){
            if(numberIsPrime(i)){
                this.resultado += i;
            }
        }
    }
}
```

```
public long getResultado()  
{  
    return this.resultado;  
}  
  
public void detener() {  
    this.bandera = false;  
}  
}
```

Ventana.java

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    String s = jButton1.getText();
    if (s.compareTo("Push") == 0){
        long t1 = System.currentTimeMillis();

        String entrada = jTextField1.getText();
        int nHilos = Integer.parseInt(entrada);
        System.out.println("Número de hilos: " + entrada);

        int numsPerHilo = 5000000 / nHilos;

        try{
            MiHilo[] hilos = new MiHilo[nHilos];

            for(int i=0; i < nHilos; i++){
                hilos[i] = new MiHilo(i, 1000 * i, true, i * numsPerHilo + 1, (i + 1) * numsPerHilo +
1);
                hilos[i].start();
            }

            long resultadoFinal = 0;
            for(int i=0; i < nHilos; i++){
                hilos[i].join();
                resultadoFinal += hilos[i].getResultado();
            }

            System.out.println("Resultado final: " + resultadoFinal);
        }catch (NumberFormatException nfe) {
            System.out.println(nfe);
        }catch (InterruptedException ie){
            System.out.println(ie);
        }
        long t2 = System.currentTimeMillis();
        System.out.println("Tiempo final: " + (t2 - t1) + " milisegundos");
    }
}
```

Evidencia de pruebas

The screenshot shows an IDE with a Java program. The code is as follows:

```
90 for(int i=0; i < nHilos; i++){
91     hilos[i].join();
92     resultadoFinal += hilos[i].getResultado();
93 }
94
95 System.out.println("Resultado final: " + resultadoFinal);
96 }catch (NumberFormatException nfe ){
97     System.out.println(nfe);
98 }catch (InterruptedException ie){
99     System.out.println(ie);
100 }
101 long t2 = System.currentTimeMillis();
102 System.out.println("Tiempo final: " + (t2 - t1) + " milisegundos");
103 }
104
105
106 /**
107  * @param args the command line arguments
108  */
```

On the right, a small window titled "Ventana" contains a text field with the value "1" and a "Push" button. Below the code editor, the "Output - ClaseHilos (run)" window displays the following output:

```
run:
Número de hilos: 1
Hilo 0 empieza
Resultado final: 838596693108
Tiempo final: 3229 milisegundos
```

The screenshot shows the same IDE with the same Java program, but the "Ventana" window now has the value "10" in the text field. The "Output - ClaseHilos (run)" window displays the following output:

```
run:
Número de hilos: 10
Hilo 0 empieza
Hilo 1 empieza
Hilo 5 empieza
Hilo 3 empieza
Hilo 4 empieza
Hilo 2 empieza
Hilo 6 empieza
Hilo 7 empieza
Hilo 8 empieza
Hilo 9 empieza
Resultado final: 838596693108
Tiempo final: 708 milisegundos
```

```
91         hilos[i].join();
92         resultadoFinal += hilos[i].getResultado();
93     }
94
95     System.out.println("Resultado final: " + resultadoFinal);
96     }catch (NumberFormatException nfe ){
97         System.out.println(nfe);
98     }catch (InterruptedException ie){
99         System.out.println(ie);
100     }
101     long t2 = System.currentTimeMillis();
102     System.out.println("Tiempo final: " + (t2 - t1) + " milisegundos");
103 }
104
105
106 /**
107  * @param args the command line arguments
108  */
```

Ventana > jButton1ActionPerformed > if (s.compareTo("Push") == 0) > try > resultadoFinal >

Output - ClaseHilos (run) X

Hilo 19 empieza
Hilo 12 empieza
Hilo 24 empieza
Hilo 27 empieza
Hilo 20 empieza
Hilo 80 empieza
Hilo 26 empieza
Hilo 46 empieza
Hilo 23 empieza
Hilo 17 empieza
Hilo 90 empieza
Hilo 37 empieza
Resultado final: 838596693108
Tiempo final: 551 milisegundos

100

Push

```
91         hilos[i].join();
92         resultadoFinal += hilos[i].getResultado();
93     }
94
95     System.out.println("Resultado final: " + resultadoFinal);
96     }catch (NumberFormatException nfe ){
97         System.out.println(nfe);
98     }catch (InterruptedException ie){
99         System.out.println(ie);
100     }
101     long t2 = System.currentTimeMillis();
102     System.out.println("Tiempo final: " + (t2 - t1) + " milisegundos");
103 }
104
105
106 /**
107  * @param args the command line arguments
108  */
```

Ventana > jButton1ActionPerformed > if (s.compareTo("Push") == 0) > try > resultadoFinal >

Output - ClaseHilos (run) X

Hilo 601 empieza
Hilo 738 empieza
Hilo 734 empieza
Hilo 534 empieza
Hilo 576 empieza
Hilo 556 empieza
Hilo 555 empieza
Hilo 519 empieza
Hilo 471 empieza
Hilo 448 empieza
Hilo 428 empieza
Hilo 423 empieza
Hilo 415 empieza
Resultado final: 838596693108
Tiempo final: 595 milisegundos

1000

Push

```
91         hilos[i].join();
92         resultadoFinal += hilos[i].getResultado();
93     }
94
95     System.out.println("Resultado final: " + resultadoFinal);
96     catch (NumberFormatException nfe ){
97         System.out.println(nfe);
98     } catch (InterruptedException ie){
99         System.out.println(ie);
100     }
101     long t2 = System.currentTimeMillis();
102     System.out.println("Tiempo final: " + (t2 - t1) + " milisegundos");
103 }
104 }
105
106 /**
107  * @param args the command line arguments
108  */
```

Ventana > jButton1ActionPerformed > if (s.compareTo("Push") == 0) > try > resultadoFinal >

Output - ClaseHilos (run) X

Hilo 9955 empieza
Hilo 9956 empieza
Hilo 9967 empieza
Hilo 9907 empieza
Hilo 9987 empieza
Hilo 9929 empieza
Hilo 9927 empieza
Hilo 9990 empieza
Hilo 9998 empieza
Hilo 9995 empieza
Hilo 9996 empieza
Hilo 9942 empieza
Hilo 9947 empieza
Resultado final: 838596693108
Tiempo final: 1809 milisegundos

10000

Push

Output - ClaseHilos (run) X

Hilo 856 empieza
Hilo 855 empieza
Hilo 854 empieza
Hilo 853 empieza
Hilo 852 empieza
Hilo 851 empieza
Hilo 850 empieza
Hilo 849 empieza
Hilo 848 empieza
Hilo 846 empieza
Hilo 845 empieza
Hilo 844 empieza
Hilo 847 empieza
Resultado final: 838406702854
Tiempo final: 681 milisegundos

1500

Push