**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 *places* 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:

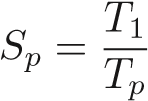
**Algoritmo para determinar si** *n* **es un número primo.** Devuelve *verdadero* o *falso*.

1. Si *n* es menor que 2, el algoritmo termina devolviendo *falso*.
2. Para *i* desde 2 hasta ⌈√*n* ⌉, 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*.

1. 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:



En donde:

* *p* es el número de procesadores (o núcleos).
* *T*1 es el tiempo que tarda en ejecutarse la versión secuencial del programa.
* *Tp* es el tiempo que tarda en ejecutarse la versión paralela del programa utilizando *p* procesadores.
* *Sp* 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
   1. S10 = 3200 ms / 708 ms = 4.52
3. 100 Hilos – 551 milisegundos
   1. S100 = 3200 ms / 551 ms = 5.80
4. 1000 Hilos – 595 milisegundos
   1. S1000 = 3200 ms / 595 ms = 5.37
5. 1500 Hilos – 681 milisegundos
   1. S1500 = 3200 ms / 681 ms = 4.69
6. 10000 Hilos – 1009 milisegundos
   1. S10000 = 3200 ms / 1009 ms = 3.17

Entre 20 y 100 hilos es lo más optimo 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**

