UNIVERSIDAD DEL VALLE DE GUATEMALA

CC3039- Computación Paralela y Distribuida

Sección 10

Ing. Sebastián Galindo

Laboratorio 1

SEBASTIAN ARISTONDO PEREZ 20880

JOSE DANIEL GONZALEZ CARRILLO 20293

Ejercicio 1

a)

función más importante

```
double sum = 0.0;
for (int k = 0; k < n; k++) {
    sum += factor/(2*k+1);
    factor = - factor;
}
double pi_approx = 4.0*sum;
```

Versión secuencial

```
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesSeq
pi_approx =    3.1414926535900345e+00
```

Versión paralela

t, N	Resultado
5, 1000	**3.1405926538397924e+00**
10, 10000	**3.1414926535900443e+00**
20, 20000	**-3.1401200312155484e+00**
30, 30000	**3.1522569711692969e+00**
40, 50000	**3.1393750929644750e+00**

A medida que se modifica la cantidad de Threads y la cantidad de iteraciones, se puede observar que el valor va oscilando con respecto al valor real. Se puede observar que incluso hay un valor negativo. Es decir, con este programa la aproximación no es buena.

```
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesNaive 5 1000
threads = 5
pi_approx:  3.1405926538397924e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesNaive 10 10000
threads = 10
pi_approx:  3.1414926535900443e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesNaive 20 20000
threads = 20
pi_approx:  -3.1401200312155484e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesNaive 30 30000
threads = 30
pi_approx:  3.1522569711692969e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesNaive 40 50000
threads = 40
pi_approx:  3.1393750929644750e+00
```

b)

Esta es una dependencia verdadera o dependencia de input/output, porque otras iteraciones dependen del valor que tenga el factor previo, si no, es posible que hagan una suma en la serie, cuando en realidad tocaba hacer una resta. Esto se puede catalogar también como una race condition, porque varios hilos pueden cambiar el valor de factor a la vez y este será no determinista.

c)

La razón por la que factor = - factor es porque en la serie se puede observar que las sumas y restas se alternan. Es decir, en la iteración 0 se suma, en la iteración 1 se resta y así sucesivamente. Entonces por esto es necesario cambiar el factor por su negativo.

d)
función más importante

```
for (int k = 0; k < n; k++) {
    if (k % 2 == 0){
        factor = 1.0;
    }
    else {
        factor = -1.0;
    }
    sum += factor/(2*k+1);
}
```

t, N	Resultado
5, 1000000	3.1405926538397924e+00
10, 1000000	3.1414926535900447e+00
20, 2000000	3.1415426535898257e+00
30, 3000000	3.1415593202564707e+00
40, 5000000	3.1415726535897948e+00

A diferencia del inciso a, donde el valor se alejó del original, acá se puede observar que con más iteraciones, más se acercó al original.

```
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesNaiveCopy 5 1000000
threads = 5
pi_approx:  3.1405926538397924e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesNaiveCopy 10 1000000
threads = 10
pi_approx:  3.1414926535900447e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesNaiveCopy 20 2000000
threads = 20
pi_approx:  3.1415426535898257e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesNaiveCopy 30 3000000
threads = 30
pi_approx:  3.1415593202564707e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesNaiveCopy 40 5000000
threads = 40
pi_approx:  3.1415726535897948e+00
```

e)
función más importante

```
for (int k = 0; k < n; k++) {
    if (k % 2 == 0){
        factor = 1.0;
    }
    else {
        factor = -1.0;
    }
    sum += factor/(2*k+1);
}
```

Intento	Resultado
1	3.1414926535900345e+00
2	3.1414926535900345e+00
3	3.1414926535900345e+00
4	3.1414926535900345e+00
5	3.1414926535900345e+00

Las diferentes iteraciones no tienen mayor cambio ya que al utilizar un único hilo no existe race condition con la variable factor por lo tanto la suma es de forma ordenada.

```
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesNaiveCopy 1 1000000
threads = 1
pi_approx:  3.1414926535900345e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesNaiveCopy 1 1000000
threads = 1
pi_approx:  3.1414926535900345e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesNaiveCopy 1 1000000
threads = 1
pi_approx:  3.1414926535900345e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesNaiveCopy 1 1000000
threads = 1
pi_approx:  3.1414926535900345e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesNaiveCopy 1 1000000
threads = 1
pi_approx:  3.1414926535900345e+00
```

f)

función más importante

```
#pragma omp parallel for num_threads(thread_count) schedule(guided, 128) reduction(+:sum) private(factor)
for (int k = 0; k < n; k++) {
    if (k % 2 == 0){
        factor = 1.0;
    }
    else {
        factor = -1.0;
    }
    sum += factor/(2*k+1);
}
```

t, N	Resultado
5, 1000000	3.1415916535897614e+00
10, 1000000	3.1415916535897197e+00
20, 2000000	3.1415921535897193e+00
30, 3000000	3.1415923202563865e+00
40, 5000000	3.1415924535897277e+00

Ahora que ya no existe race condition con la variable factor cuando incrementamos n y el número de threads, más cerca es la aproximación al valor real de pi. Es importante destacar que cada vez se acerca más por los valores más pequeños los primeros 5 decimales ya son los correctos en todos los intentos

```
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ gcc -o piSeriesNaiveCopy  piSeriesNaive_copy.c -fopenmp
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesNaiveCopy 1000000 5
threads = 5
pi_approx =   3.1415916535897614e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesNaiveCopy 1000000 10
threads = 10
pi_approx =   3.1415916535897197e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesNaiveCopy 2000000 20
threads = 20
pi_approx =   3.1415921535897193e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesNaiveCopy 3000000 30
threads = 30
pi_approx =   3.1415923202563865e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesNaiveCopy 5000000 40
threads = 40
pi_approx =   3.1415924535897277e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$
```

g)
función más importante

```
#pragma omp parallel for num_threads(thread_count) schedule(guided, 128) reduction(+:sum) private(factor)
for (int k = 0; k < n; k++) {
    if (k % 2 == 0){
        factor = 1.0;
    }
    else {
        factor = -1.0;
    }
    sum += factor/(2*k+1);
}
```

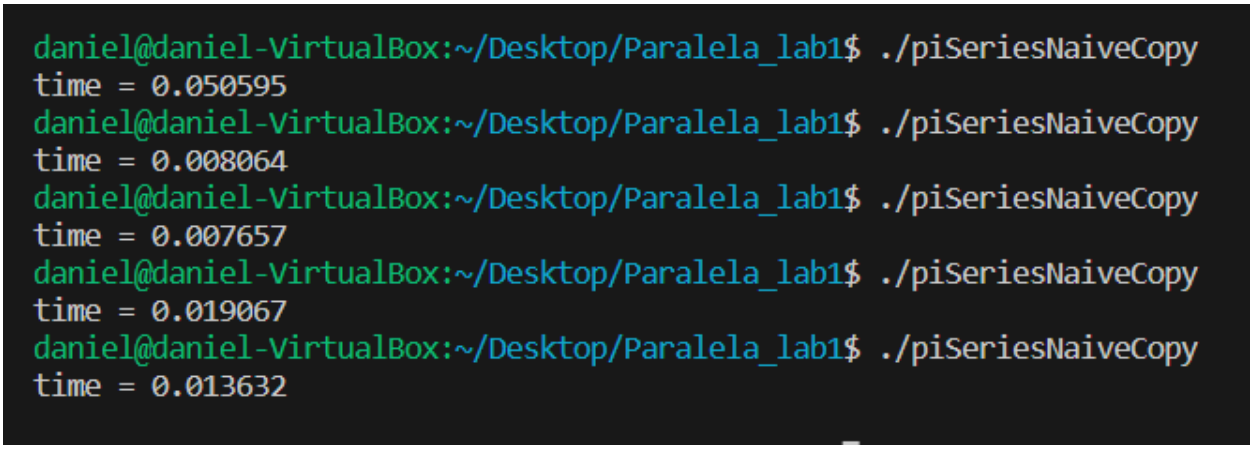
	Tiempo secuencial en segundos	Tiempo paralelo en segundos (threads = n_cores)	Tiempo paralelo en segundos (threads = 2 * n_cores)	Tiempo paralelo en segundos (n = n*10 y threads = cores)
	0.002040	0.005225	0.003640	0.005821
	0.002155	0.005151	0.004246	0.012727
	0.002295	0.000889	0.004132	0.037248
	0.002129	0.003026	0.003157	0.016081
	0.002195	0.004569	0.003119	0.038687
Promedio	0.0021628	0.003772	0.0036588	0.0221128

```
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesNaiveCopy 10 1000000
time = 0.00204
pi_approx:  3.1414926535900345e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesNaiveCopy 10 1000000
time = 0.002155
pi_approx:  3.1414926535900345e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesNaiveCopy 10 1000000
time = 0.002295
pi_approx:  3.1414926535900345e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesNaiveCopy 10 1000000
time = 0.002129
pi_approx:  3.1414926535900345e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesNaiveCopy 10 1000000
time = 0.002195
pi_approx:  3.1414926535900345e+00
```

	Tiempo paralelo en segundos (threads = n_cores)	Tiempo paralelo en segundos (threads = 2 * n_cores)	Tiempo paralelo en segundos (n = n*10 y threads = cores)
Speedup	0.57	0.59	N/A
Eficiencia	0.1425	0.07375	N/A
Escalabilidad fuerte	N/A	0.2956	N/A
Escalabilidad débil	N/A	N/A	0.09

h)

Intento	**auto**
1	0.050595
2	0.008064
3	0.007657
4	0.019067
5	0.013632
Promedio	0.019803



Block size 16

Intento	Static	Dynamic	guided
1	0.045162	0.048852	0.045824
2	0.028190	0.059824	0.019402
3	0.009209	0.058598	0.006259
4	0.026089	0.038209	0.016339
5	0.008552	0.105955	0.015208
Promedio	0.0234404	0.06229	0.0206

Block size 64

Intento	Static	Dynamic	guided
1	0.009782	0.051938	0.057707
2	0.058745	0.044188	0.031037
3	0.046486	0.056062	0.040247
4	0.052098	0.009308	0.029226
5	0.039115	0.038026	0.019029
Promedio	0.0412452	0.0399	0.03545

Block size 128

Intento	Static	Dynamic	guided
1	0.014850	0.015682	0.011817
2	0.006537	0.021797	0.007010
3	0.015830	0.018847	0.007376
4	0.019026	0.018059	0.020069
5	0.012223	0.046261	0.016335
Promedio	0.0136932	0.02413	0.0125214

	Speedup
Auto	0.1092
Static (16)	0.0922
Static (64)	0.0524
Static (128)	0.1579
Dynamic (16)	0.0347
Dynamic (64)	0.0542
Dynamic (128)	0.0896
Guided (16)	0.1050
Guided (64)	0.0610
Guided (128)	0.1727

Los mejores resultados se obtuvieron con la política de **guided** con block size de 128.

Ejercicio 2

a)
función más importante

```
#pragma omp parallel for num_threads(thread_count) reduction(+:sum)
for (int k = 0; k < n; k++) {
    if (k % 2 == 0){
        sum += 1.0/(2*k+1);
    }
    else {
        sum += -1.0/(2*k+1);
    }
}
```

t, N	Resultado

5, 1000000	**3.1415916535897614e+00**
10, 1000000	**3.1415916535897197e+00**
20, 2000000	**3.1415921535897198e+00**
30, 3000000	**3.1415923202563860e+00**
40, 5000000	**3.1415924535897282e+00**

Ahora tomaremos medidas del mejor resultado de pi para comparar, por lo tanto usaremos t = 40 y N = **5000000.** A continuación las 5 tomas

Intento	**Tiempo**
1	0.014189
2	0.017606
3	0.018707
4	0.036393
5	0.016236
Promedio	0.0206
Speedup	0.1050

Como podemos ver el Speedup de Guide con block sixe de 128 obtuvo un speedup de 0.1727. mientas que esta implementación la cual logramos quitar una instrucción obtuvo un speedup de 0.1050..

```
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesAlt 40 5000000
threads = 40
time = 0.014189
pi_approx:  3.1415924535897282e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesAlt 40 5000000
threads = 40
time = 0.017606
pi_approx:  3.1415924535897282e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesAlt 40 5000000
threads = 40
time = 0.018707
pi_approx:  3.1415924535897282e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesAlt 40 5000000
threads = 40
time = 0.036393
pi_approx:  3.1415924535897282e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesAlt 40 5000000
threads = 40
time = 0.016236
pi_approx:  3.1415924535897282e+00
```

b)

función más importante

```
#pragma omp parallel for num_threads(thread_count) reduction(+:sum)
for (int k = 0; k < n; k++) {
    if (k % 2 == 0){
        sum += 1.0/(2*k+1);
    }
    else {
        sum += -1.0/(2*k+1);
    }
}
```

Con la directiva de compilación -O2 para el modelo con schedule guided y block size 128.

Intento	Tiempo
1	0.019335
2	0.009088
3	0.004419
4	0.013830
5	0.006634
Promedio	0.0107
Speedup	0.2021

Pudimos observar que el tiempo promedio con la bandera de optimización se reduce considerablemente, por lo tanto esto significa que el speedup también tuvo un cambio significativo de 0.3. Durante todo el laboratorio en cada ejercicio se buscó que el programa fuera lo más eficiente posible. Antes de aplicar la bandera el código se había logrado llevar a la solución más óptima posible a nivel de código. Pero al utilizar la bandera la optimización se realizado a nivel de código máquina, brindándonos el mejor desempeño de la aproximación de pi. Según leímos, la bandera permite realizar optimizaciones a los ciclos, así como al flujo de control. Esto fue en parte lo que permitió reducir aún más el tiempo promedio y mejorar un poco más el speed up.

```
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesAlt 40 5000000
threads = 40
time = 0.019335
pi_approx:  3.1415924535897282e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesAlt 40 5000000
threads = 40
time = 0.009088
pi_approx:  3.1415924535897282e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesAlt 40 5000000
threads = 40
time = 0.004419
pi_approx:  3.1415924535897282e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesAlt 40 5000000
threads = 40
time = 0.01383
pi_approx:  3.1415924535897282e+00
daniel@daniel-VirtualBox:~/Desktop/Paralela_lab1$ ./piSeriesAlt 40 5000000
threads = 40
time = 0.006634
pi_approx:  3.1415924535897282e+00
```