

# Cálculo número Pi

David Gil Bautista

Grupo D3

Para la solución del problema de aproximación de una integral con  $n$  procesos mi implementación ha sido la siguiente:

```
calculopi.cpp  x  fun tiempo.h  x  fun tiempo.c  x
46 // -----
47 // función que ejecuta cada hebra
48
49 void * funcion_hebra( void * ih_void )
50 {
51     unsigned long ih = (unsigned long) ih_void ; // número o índice de esta hebra
52     double sumap = 0.0 ;
53
54     // calcular suma parcial en "sumap"
55     for (double i = ih; i < m; i+=nhebras ){
56         sumap += f((i+0.5)/m);
57     }
58
59     resultado_parcial[ih] = sumap ; // guardar suma parcial en vector.
60     return NULL ;
61 }
62 // -----
63 // cálculo concurrente
64
65 double calcular_integral_concurrente( )
66 {
67     // crear y lanzar $n$ hebras, cada una ejecuta "funcion\concurrente"
68
69     double sumaTotal = 0.0;
70     pthread_t hebras[nhebras];
71
72     for (int i = 0; i < nhebras; ++i)
73         pthread_create(&(hebras[i]), NULL, funcion_hebra, (void *)i);
74
75     // esperar (join) a que termine cada hebra, sumar su resultado
76
77     for (int i = 0; i < nhebras; ++i)
78         pthread_join(hebras[i], NULL);
79
80     // devolver resultado completo
81
82     for (int i = 0; i < nhebras; ++i)
83         sumaTotal += resultado_parcial[i];
84
85     return sumaTotal/m ;
86 }
87
88
89 // -----
90
```

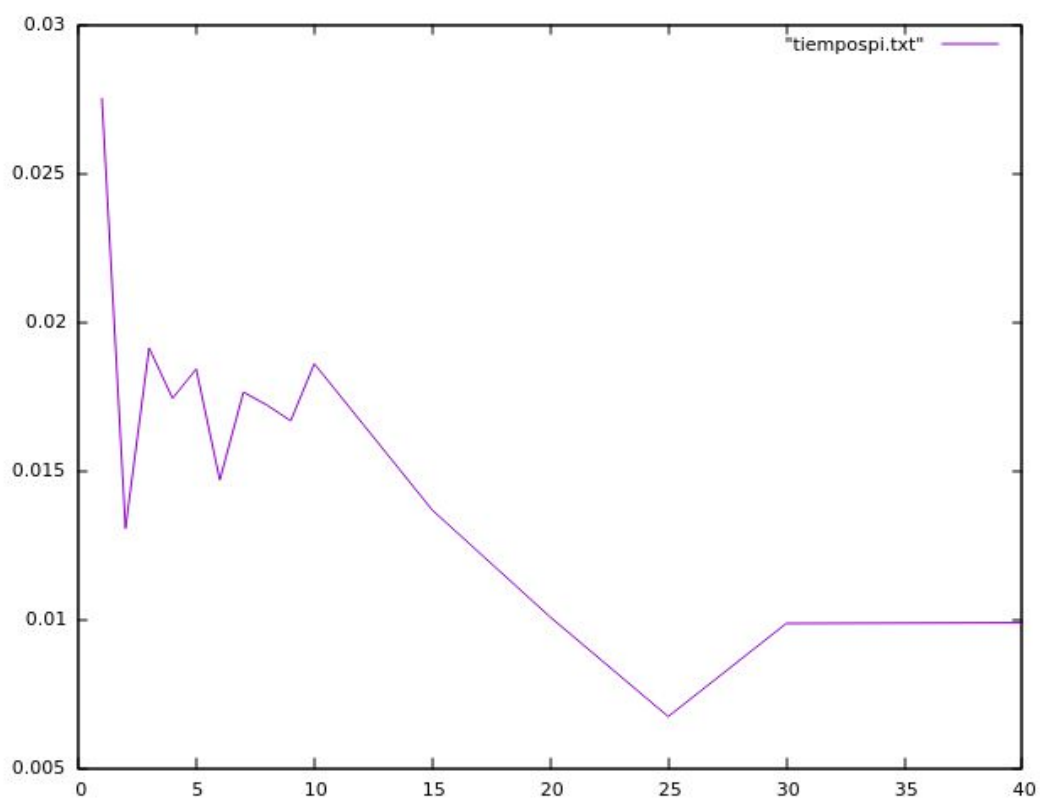
Line 123, Column 18

Cada hebra ejecuta la función *funcion\_hebra* que recibe como parámetro **ih\_void**, esta será la identidad de cada hebra. Una vez llamada la función se ejecuta un bucle que comenzará desde la *ih* recibida e itera *muestras/hebras* veces.

El resultado de la integral parcial se guardará en *resultado\_parcial*, que es un array con mismo número de elementos que hebras.

La función *calcular\_integral\_concurrente* realiza la creación de las *n* hebras, su inicialización y recogida de datos parciales.

A continuación se muestra una gráfica con el tiempo de ejecución para distintas hebras.



Mediante la función de cálculo secuencial sale un tiempo similar al de la ejecución concurrente ya que no se divide el trabajo.